[redacted]

UIUCDCS-R-75-698

PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS

(Reference Manual of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)

by

J.N. Culliney

February 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

117

UIUCDCS-R-75-698

PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS (Reference Manual of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)

by
J.N. Culliney

February 1975

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

ABSTRACT

This paper explains the operation and usage of two FORTRAN computer programs, NETTRA-G1 and NETTRA-G2, developed for NOR-network transduction (transformation and reduction).

Existing (non-optimal) NOR-gate networks and their required output functions are given to the programs as input. The programs, in general, add, change, and/or delete connections in the network in an effort to reduce the cost of the network (defined in terms of numbers of gates and connections) as much as possible. Gates are examined individually; their input connections and potential input connections are evaluated under certain conditions of connectability and disconnectability in order to effect the changes in network configuration and thus reduce network cost.

These programs are only two out of a whole system of programs, designated by the name "NETTRA" (for NETwork TRAnsduction), which implement different NOR-network transduction procedures.

The theoretical basis for the algorithms implemented by NETTRA-G1 and -G2 is detailed in earlier reports ([1] and [2]).

ACKNOWLEDGMENT

The author is greatly appreciative of Prof. S. Muroga for his discussions and guidance relating to the preparation of this paper, and also, for his careful reading and valuable suggestions for the improvement of the original manuscript. The author is also indebted to H.C. Lai and Y. Kambayashi upon whose related research much of the work reported herein depends.

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

TABLE OF CONTENTS

1. INTRODUCTION

This manual is intended to instruct the reader in the use of the FORTRAN programs 'NETTRA-G1' and 'NETTRA-G2'. These programs realize the algorithms described in detail in [1], and this manual will assume a knowledge of those definitions and algorithms in [1] as well as a knowledge of the general description of network transduction procedures presented in [2].

NETTRA-G1 and -G2 are only two out of a whole system of programs developed at the University of Illinois by the research group led by Prof. S. Muroga. The generic name 'NETTRA' (for NETwork TRAnsduction) designates the whole collection of programs comprising the system. All of the programs in the NETTRA system either transform or assist in transforming networks of interconnected NOR gates realizing various functions of their respective sets of input variables. By these transformations, a large, non-optimal network of NOR gates realizing one or more various functions can often be <u>reduced</u> to a smaller, less expensive (in terms of the number of required gates and interconnections, for example), near-optimal network realizing the same functions(s). In general, such a transformation could involve a complete reorganization of the network: the addition and/or deletion of gates; the addition and/or deletion of connections among gates; and/or the substitution of certain connections for various others. The procedures realized by NETTRA-G1 and NETTRA-G2 can accomplish any of these changes, with the exception of adding gates to the network.

The procedures realized in the programs NETTRA-G1 and -G2 are more complex than those appearing in [3] and require more computer time to execute. However, they are more powerful also, and they can often reduce

a network when it is impossible to do so by those other procedures.

The programs, NETTRA-PG1, -P1, and -P2 described in [3], though, are more efficient than NETTRA-G1 and -G2 when first applied to large, far-from-optimal networks. NETTRA-G1 and -G2 are most useful when applied to more nearly optimal networks where it is fairly difficult to achieve a further reduction of the network.

As can be seen later, NETTRA-G1 and -G2 are quite similar programs, sharing the same major subroutine in fact; but they realize significantly different transformation procedures.

The next two sections, Sections 2 and 3, discuss the two programs in greater detail and present some examples of the effectiveness of their transformations. This is followed, in Section 4, by a description of the functions of the subroutines which support the subroutines actually realizing the procedures. Section 5 outlines the preparation of input for the two programs. Finally, in the appendix, a complete listing of each of the FORTRAN programs, NETTRA-G1 and NETTRA-G2 is given.

2. A GENERAL NETWORK TRANSDUCTION PROCEDURE TO REMOVE NON-SPECIFIC GATES

This and the following section will discuss, respectively, the NOR-network "transduction" (transformation and reduction) procedures realized by the FORTRAN programs designated NETTRA-G1 and NETTRA-G2. These programs realize procedures which are strictly network reduction procedures. In other words, when they are applied in an attempt to transform a network, the cost of the network will be either reduced or unchanged - it will never be increased.

The input to either of these programs is a description of a particular NOR network under consideration. This description (explained in detail in Section 5) consists of a set of various network parameters. The output of both programs is a description of the "transformed" network (if a transformation was possible).

As opposed to the procedure to be discussed in Section 3, the procedure discussed in this section does not attempt to remove specific gates from a network. A calculation is made, during which, usually, the network connection pattern is altered and unnecessary gates are recognized and removed from the network. Due to certain "orderings" which are needed in the procedure to perform selection decisions, there is a definite "preference" to remove certain gates from the network rather than certain others; but the procedure definitely does not focus its power on attempting the removal of a specific gate.

As a by-product of the calculation, compatible sets of permissible functions† are generated. This information is necessary for the further

---

† Refer to [1] for definitions of unfamiliar terminology.

application described in Section 2.2 of [3].

The entire NETTRA-G1 program requires 144 K bytes of core storage, about 59 K being occupied by the actual program instructions and about 85 K by the stored data (compiled by FORTRAN H (OPT 2) compiler).

The following subroutines, written in FORTRAN IV for the IBM 360/75, consititute the program NETTRA-G1: MAIN, PROCII, CONCCO, ELANDO, MINI2, RNONES, SUBNET, and OUTPUT. Two system-supplied timing routines, STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without harming the procedure itself. The functions of the support subroutines, MAIN, CONCCO, ELANDO, MINI2, RNONES, SUBNET, and OUTPUT, will be discussed in Section 4.

The general organization of the program NETTRA-G1 is shown in Fig. 2.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

2.1 Flowchart of the Subroutine Realizing the Procedure

Although NETTRA-G1 is composed of eight subroutines, the logic realizing the transduction procedure is essentially embodied in just one subroutine: PROCII (for PROCedure II, a name assigned during the development of the program). The following discussion of PROCII will assume a general knowledge of the information contained in [1].

Explanations of the purposes of the variables and arrays appearing in the subroutine can be found in the program listing of PROCII in the appendix. It is, however, convenient to define some of the variables at this point in order to discuss the flowchart of PROCII which appears in Figs. 2.1.1 and 2.1.2.

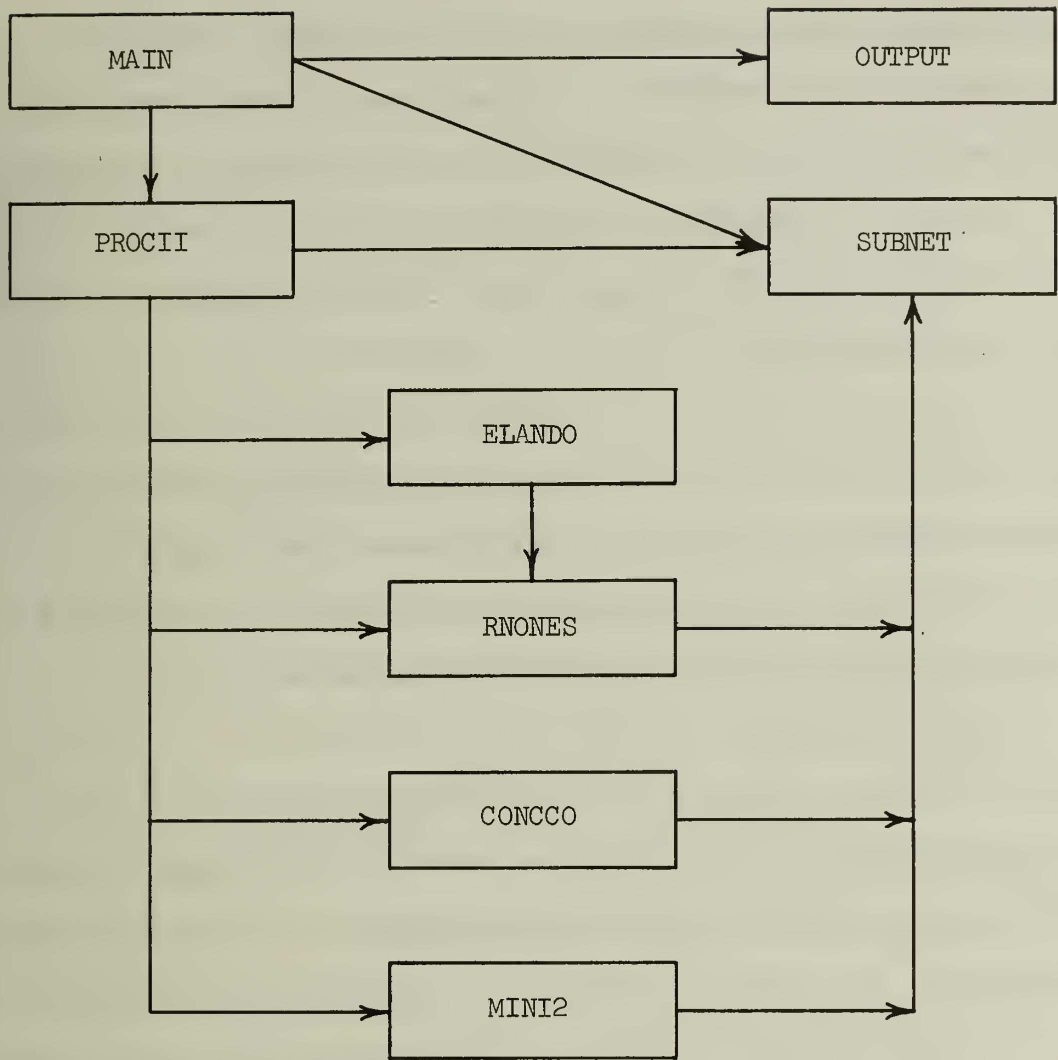


Fig. 2.1 General organization of the program NETTRA-G1.

N is the number of external variables, n, if only uncomplemented variables are allowed as inputs. If both complemented and uncomplemented are available (i.e., n variables and their n complements) then N is equal to 2n. Note that this is strictly the representation internal to the program; for input-output purposes (as described in Section 5) N and n are always equal.

R is the number of gates specified by the input data to the program. It includes all gates declared to be present by the input data, even though some of them may be isolated (i.e., not connected to other gates in the network). Internally, the program represents the gates 1, 2, ..., R by the labels N + 1, N + 2, ..., N + R. (External variables are labeled 1, 2, ..., N internally.)

NR is equal to the sum N + R. It is often convenient to treat both external variables and gates in a similar manner. External variables being labeled 1, 2, ..., N and gates being labeled N + 1, ..., N + R (internally), the number N + R is frequently required.

GORDER is an array containing a certain ordering of the various gate and external variable labels. In other words, in the locations GORDER(1), GORDER(2), ..., GORDER(NR) are stored the numbers 1, 2, ..., NR in a certain order. The ordering represented by the array GORDER satisfies the following criterion: for every gate or external variable, i, which feeds another gate, j, gate j precedes gate i in the ordering (for this to be possible, the network is assumed to be loop-free).

GSMALL is a two-dimensional array used to store the intermediate and final calculated compatible sets.† GSMALL entries are initialized to "don't-cares" at the beginning of the procedure; upon termination of the algorithm, the determined compatible sets can be read directly from GSMALL.

† For simplicity, sometimes just the words "compatible sets" will be used to denote compatible sets of permissible functions.

GSMALL(i, j) contains (or rather, will contain by the end of the procedure) the j^{th} component of the vector representing the compatible set of permissible functions for gate or external variable i.

In block 1 of the flowchart (Fig. 2.1.1), an ordering of gates and external variables is determined and stored in the array GORDER. Also, at approximately this point in the program, initial values are assigned to many of the arrays and variables which will be used later.

Block 2 simply initializes a counter, the variable GCOUNT, used in the program loop consisting of blocks 3, 4, 7, 8, and 9.

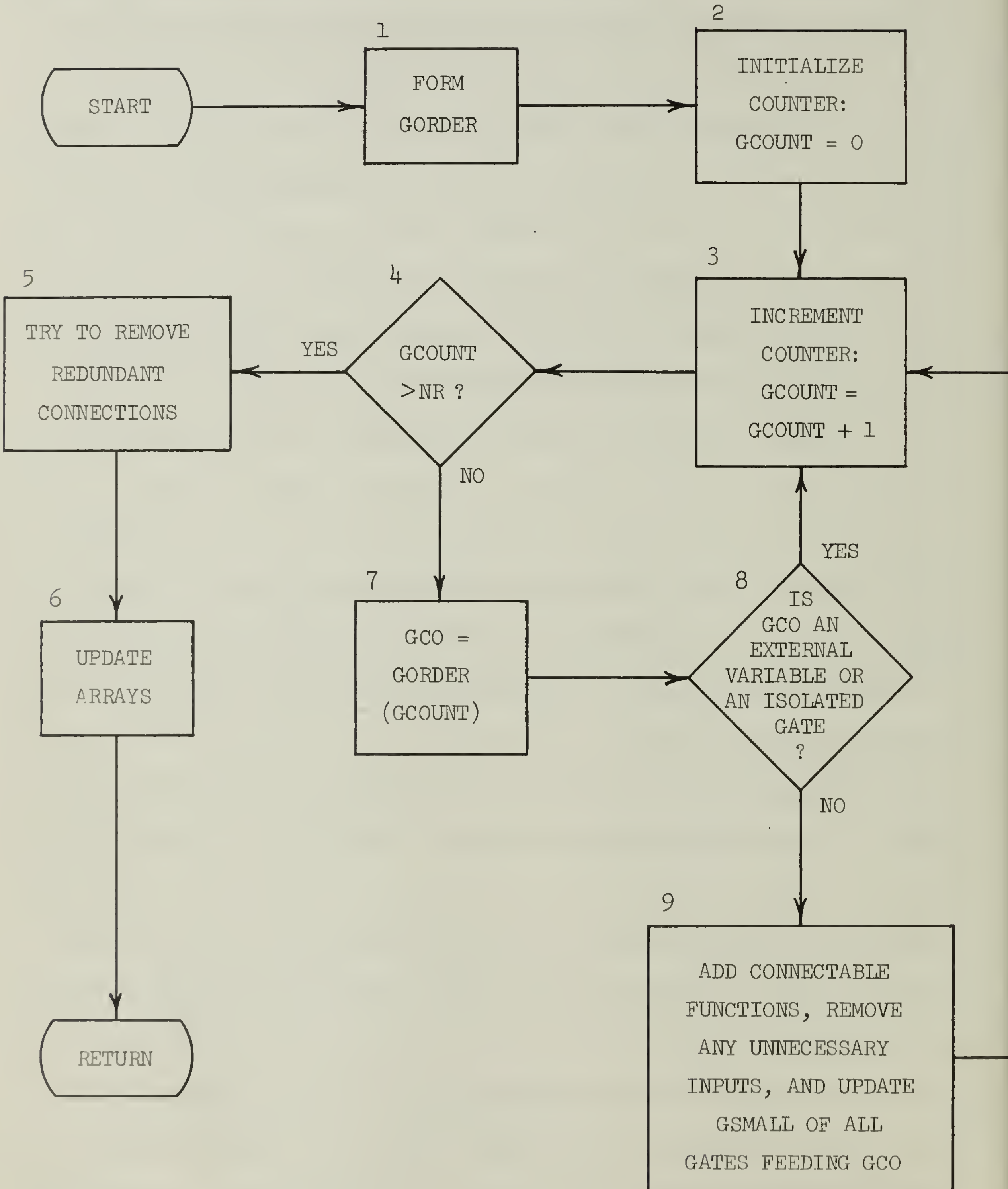
Block 3 increments the counter, GCOUNT. The program loop formed by blocks 3, 4, 7, 8, and 9 executes once for each value of GCOUNT: 1, 2, 3, ...

When GCOUNT is incremented beyond the value NR, it is detected in block 4. This is a sign that the algorithm has finished, having scanned every gate in the network, and the program enters block 5. Otherwise, the program proceeds to block 7.

In block 5 the algorithm has essentially finished. A subroutine (MINI2) is called which quickly searches for and removes certain redundant connections which may still remain in the network (for example, certain new connections that might have been added unnecessarily in block 9). The removal of such connections does not cause a change in the output functions of the network.

At the end of every transformation there are certain "house-keeping" chores which must be performed: updating or restoring values in arrays and variables. This task is done in block 6. This is followed by a return to the calling subroutine (MAIN).

Fig. 2.1.1 Generalized flowchart of PROCII.



In block 7, the variable GCO is assigned the label (of a gate or external variable) contained in the $(GCOUNT)^{th}$ position of the ordering stored in the array GORDER. GCO becomes the name of the gate (or external variable) about to be examined by the program.

If GCO happens to be either an external variable or an isolated gate, no action needs to be taken. In such a case block 8 sends control back to block 3. Otherwise, the program continues to block 9.

Block 9 performs many functions and is actually quite complex. So block 9 is detailed in Fig. 2.1.2 as consisting of sub-blocks 10 through 23. In these blocks the program focuses its attention on gate GCO and its inputs.

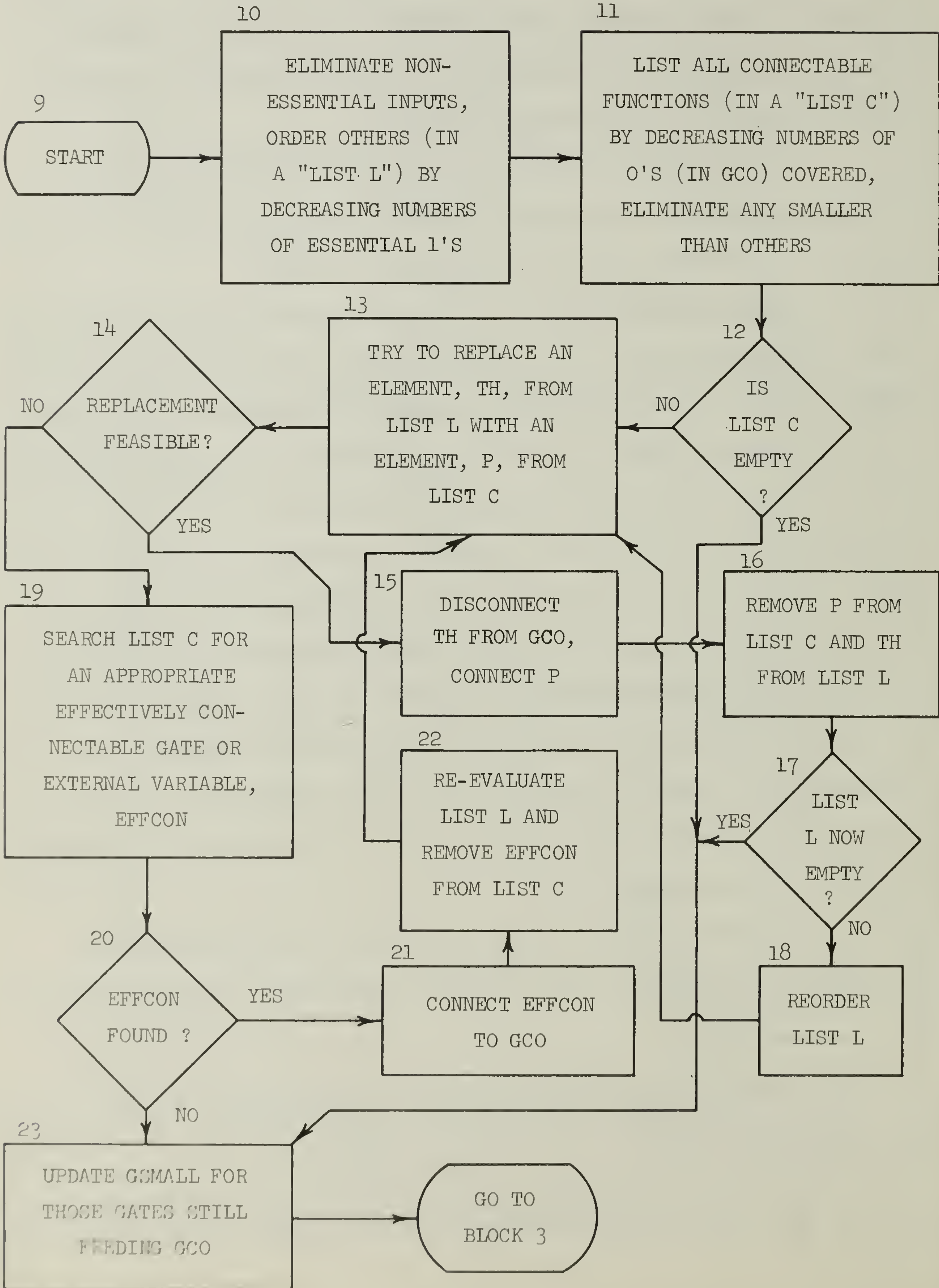
During the previous part of the calculation, the output requirement (i.e., the set of permissible functions†) of gate GCO (0, 1 or *) has already been determined for each of the 2^n possible combinations of external variable values. Whenever the required output (i.e., GSMALL component) of GCO is a 1 for a certain input vector, all of the immediate successors of GCO (i.e., the gates and/or external variables which feed GCO) must have an output of 0. Also, whenever the required output of GCO (i.e., GSMALL component) is a 0, at least one of its immediate successors must have an output of 1. If the required output of GCO is * for some input vector, the outputs of its immediate successors are unrestricted.

In block 9, it is the positions of the 0's in the permissible function vector of GCO which are of foremost importance. Due to the nature of

---

† A set of permissible functions for a gate I is represented by a 2^n-dimensional vector [stored in (GSMALL(I, J), J = 1, 2, ..., 2^n)], $(f^{(1)}, \ldots, f^{(2n)})$, where $f^{(j)} = f(x_1, x_2, \ldots, x_n)$ for $j - 1 = 2^{n-1}x_1 + 2^{n-2}x_2 + \ldots + x_n$. Each coordinate is a 0, 1, or *. If *'s are used, then the vector represents the set of all functions which are obtained by assigning 0 or 1 to each * in all possible ways.

Fig. 2.1.2 Detail of block 9 of PROCII flowchart.



NOR gates, a 1 appearing on any of the input lines to a gate will cause a 0 output of that gate. Such a 1 is called a <u>cover</u> of that 0, and the 0 is said to be <u>covered</u> by that 1. Although a 0 output may be covered by several 1's (appearing on different input lines), only a single cover is actually required to guarantee the 0 output.

<u>Block 10</u> actually represents two steps. The first step is the elimination of non-essential inputs of GCO.

Checking the immediate predecessors to GCO, the number of 1 covers for each 0 component of the permissible function vector is determined and stored. Some 0 components have only a single 1 cover. In such a case, the single 1 covering the 0 is called an <u>essential 1</u>. Any input line (connection) carrying an essential 1 for some input vector cannot be disconnected from GCO without causing the actual function of GCO to be outside the set of permissible functions for GCO. However, the input connections to GCO which do not carry essential 1's are redundant and are removed from the network in the first step of block 10. The removal of these redundant input connections must be done serially though since removing any input to a gate might cause new essential 1's to be created.

The remaining inputs to GCO all have essential 1's. They are ordered in a list L, (realized by an array named "LISTL" in the program) such that the input represented by LISTL(i) has at least as many essential 1's as the input represented by LISTL(i+1). This is the second step of block 10.

<u>In block 11</u>, first a search is made for all gates and external variables which are effectively connectable† to GCO. (The original inputs to GCO are prohibited from being considered effectively connectable to GCO.) If one of these effectively connectable functions, call it A, covers every 0 of GCO covered by another effectively connectable function, say B, then B is

---

† Refer to [1] for definitions of unfamiliar terminology.

eliminated from the collection of (effectively connectable) functions (i.e., A is greater than or equal to B in every component corresponding to a position of a 0 in the permissible function vector of GCO). Those remaining effectively connectable functions are ordered in a list C, (realized by an array named "LISTC" in the program) such that the function represented by LISTC(i) covers at least as many 0's of (the permissible function vector of) GCO as the function represented by LISTC(i+1).

If the program is unable to detect any effectively connectable functions (i.e., if list C is empty), control goes to block 23 and then to block 3. Otherwise the program proceeds to the major program loops in block 9, consisting of blocks 13 through 22. Block 12 tests for an empty list C.

Block 13 seeks an element,[†] P, from list C which can be directly substituted for an element, TH, from list L, such that the substitution would not cause the actual function of GCO to be outside its set of permissible functions. Furthermore, this TH and P are also selected in such a manner that TH is the first element in list L which can be replaced by an element in C and that P is the first element in list C which can replace that TH.

Block 14 tests if a feasible replacement has been found. If no replacement is possible, control passes to block 19. If, however, a suitable P and TH were chosen, blocks 15 through 18 perform the actual exchange of P for TH as an input of gate GCO.

First TH is disconnected from GCO in block 15. Also this block connects the new input, P, to GCO.

[†] In this case, an "element" is actually the function realized by a gate or external variable.

Block 16 removes P from the list C of effectively connectable functions and TH from the list L of inputs to GCO. If list L becomes empty by the removal of TH (block 17 test), the program has replaced all of the original inputs to GCO by new ones, and the program moves to the next step of the procedure in block 23.

Otherwise the program goes to block 18 where the elements of list L are reordered by decreasing numbers of essential 1's. This is necessary since, by the addition of P to GCO, some previously essential 1's may have become non-essential. From here, the program returns to block 13 to search for another replacement P.

Block 19 is reached when it is no longer possible to replace an element of list L with an element of list C as an input to GCO. Here, the program first searches for an element of list C which can cover at least one 0 of (the permissible function vector of) GCO which is currently covered by an essential 1 belonging to one of the original inputs to gate GCO. If such an input is found, it is assigned the label EFFCON and the program proceeds to block 21. If no EFFCON can be chosen, this implies that there can be no further replacements of original inputs to GCO by elements of list C. In this case, the program searches list C one last time looking for elements which cover at least one 0 of GCO which is currently covered only by the remaining original inputs to GCO (i.e., which is not covered by any of the newly connected functions). The group of elements satisfying this criterion are connected to GCO, and control goes to block 23.

Block 20 was discussed as part of block 19.

In block 21 the selected EFFCON is connected to GCO.

This connection requires the reordering of list L and the removal

of EFFCON from list C. This is done in block 22. The program then returns to block 13 to try again to find an element of list L which can be replaced by an element of list C. This may now be possible although it was impossible before the connection of EFFCON to GCO.

In block 23 the covering assignments are made for the gates still feeding GCO. In other words, for each 0 component of the permissible function vector of GCO, one of the gates,[†] GI (feeding GCO), producing a 1 cover for that 0 is selected. Gate GI is then required to produce a 1 output for that 0 component, and this requirement is actually a restriction on the set of permissible functions for GI. This requirement is recorded (assuming the 0 in question appears as the j^{th} component of the permissible function vector of GCO) by forcing the j^{th} component of the permissible function vector of GI to be a 1 (i.e., the value 1 is stored in the location GSMALL (GI, j)). Although other 1 covers (from other gates feeding GCO) may exist for that same 0 output, they are not "required" in the same sense as the 1 cover provided by the selected gate.

After all of the required 1 outputs of gates feeding GCO have been selected and stored (in GSMALL), the required 0 outputs of the immediate predecessors (actually, external variables feeding GCO can be, and are, ignored) are determined and stored by inserting 0's into GSMALL in the appropriate locations. It is an easy task to find the locations of these required 0's. If 1's appear in GSMALL (GCO, j_1), GSMALL (GCO, j_2), ..., GSMALL (GCO, j_k) [i.e., in the j_1^{th}, j_2^{th}, ..., j_k^{th} components of GCO's permissible function vector], then 0's must be required in GSMALL (p_1, j_1), ..., GSMALL (p_1, j_k); GSMALL (p_2, j_1), ..., GSMALL (p_2, j_k); ...;

[†] If the 0 is found to be covered by an external variable, no 1 cover is selected.

GSMALL $(p_\ell, j_1), \ldots, (p_\ell, j_k)$, where $p_1, p_2, \ldots, p_\ell$ are the labels of the immediate predecessors of GCO.

The completion of block 23 also means the completion of block 9, and execution of the program moves into block 3 of Fig. 2.1.1.

2.2 Example for NETTRA-G1

Fig. 2.2.1 shows partially the printout obtained from NETTRA-G1 for a typical example.

The original network produces a single output function and consists of 25 gates and 105 connections. Five independent, uncomplemented variables are available as inputs to the network. This information appears at the beginning of the output (Fig. 2.2.1 (a)).

This is followed by a complete truth table (b) showing the output of every gate in the original network for every possible input combination. Note it is gate 1 which realizes the output function of the network.

Next appears a description of the configuration of the network (c). Each gate is listed along with the gates and/or external variables which are its inputs. The level numbers, also to be seen in (c), will be discussed in Section 5.3.

The truth table (note that the outputs for disconnected gates are shown as all 1's) and network configuration for the transformed network resulting from the action of NETTRA-G1 are shown in (d) and (e), respectively. The derived network consists of 11 gates and 38 connections. If NETTRA-G1 were applied to this new network, a third network of 10 gates and 36 connections would be obtained.

Fig. 2.2.1 Printout for a Network Transformed by NETTRA-G1.

```
**** 5 VAR., EXAMPLE                         HEX=FF68A1F3

          NUMBER OF VARIABLES =   5
          NUMBER OF FUNCTIONS =   1
          COST COEFFICIENT A  = 100
                           B  =   1

          --- UNCOMPLEMENTED VARIABLES X ---

FUNCTION NO. 1.
   11111111011010001010000111110011

 ORIGINAL NETWORK    COST = 25105
```

(a) Heading information and network parameters.

TRUTH TABLE

X1 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2 =	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3 =	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4 =	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5 =	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 =	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	0	0	1	1
2 =	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 =	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 =	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 =	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 =	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 =	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 =	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 =	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 =	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 =	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13 =	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14 =	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16 =	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
17 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
18 =	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0
20 =	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
22 =	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
23 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
24 =	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
25 =	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0

(b) Truth table for original network.

GATE ..	LEVEL	FED BY
1	/ 1/	3 7 10 12 15 17 19 21 23 25
2	/ 3/	X1 X2 X3 X4 X5
3	/ 2/	X1 X3 X4 X5 2
4	/ 3/	X1 X2 X3
5	/ 3/	X1 X3 X4
6	/ 3/	X1 X3 X5
7	/ 2/	X1 X3 4 5 6
8	/ 3/	X1 X2 X5
9	/ 3/	X1 X4 X5
10	/ 2/	X1 X5 6 8 9
11	/ 3/	X1 X2
12	/ 2/	X1 5 8 9 11
13	/ 3/	X1 X2 X3 X4
14	/ 3/	X2 X3 X4 X5
15	/ 2/	X2 X3 X4 13 14
16	/ 3/	X2 X3 X5
17	/ 2/	X2 X3 4 13 16
18	/ 3/	X1 X2 X4
19	/ 2/	X2 X4 14 18
20	/ 3/	X1 X2 X4 X5
21	/ 2/	X2 X5 8 14 16 20
22	/ 3/	X3 X4 X5
23	/ 2/	X4 X5 9 20 22
24	/ 3/	X3 X4
25	/ 2/	X4 9 18 20 24

(c) Configuration of original network.

NETWORK DERIVED BY PROCII
TIME ELAPSED = 144 CENTISECONDS

TRUTH TABLE

X1 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2 =	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3 =	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4 =	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5 =	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 =	1	1	1	1	1	1	1	1	0	1	1	0	1	0	0	0	1	0	1	0	0	0	0	1	1	1	1	1	0	0	1	1
2 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3 =	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9 =	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
11 =	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 =	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
14 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16 =	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
17 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
18 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
22 =	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
23 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24 =	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
25 =	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0

(d) Truth table for transformed network.

GATE ..	LEVEL	FED BY
1	/ 1/	3 12 17 21 25
2	/ 1/	
3	/ 2/	X1 X3 X4 X5 11
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 1/	
9	/ 3/	X1 X4 X5
10	/ 1/	
11	/ 3/	X1 X2
12	/ 2/	X1 9 11 16 24
13	/ 1/	
14	/ 1/	
15	/ 1/	
16	/ 3/	X3 X5
17	/ 2/	X2 X3 11 22
18	/ 1/	
19	/ 1/	
20	/ 1/	
21	/ 2/	X2 X5 11 16 24
22	/ 3/	X5
23	/ 1/	
24	/ 3/	X3 X4
25	/ 2/	X4 9 11 24

* A NETWORK DERIVED BY PROCII
COST = 11038.

(e) Configuration of transformed network.

3. A SPECIALIZED NETWORK TRANSDUCTION PROCEDURE TO REMOVE A SPECIFIC GATE

The procedure about to be discussed here attempts to remove specific gates from a network, in contrast to the procedure just described. Actually, this procedure, realized by the program NETTRA-G2, consists of many applications of a smaller procedure which attempts (upon each application) to remove a specific gate from the network.

NETTRA-G2 consists of the following subroutines written in FORTRAN IV for the IBM 360/75: MAIN, PROCIV, PROCII, MINI2, CONCCO, ELANDO, RNONES, SUBNET, and OUTPUT. Two system-supplied timing routines, STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without harming the procedure itself. The functions of the support subroutines MAIN, CONCCO, ELANDO, MINI2, RNONES, SUBNET, and OUTPUT will be discussed later in Section 4.

The entire NETTRA-G2 program requires 146 K bytes of core storage, about 61 K being occupied by the actual program instructions and about 85 K by the stored data (compiled by FORTRAN H (OPT 2) compiler).

Fig. 3.1 shows the general organization of the program NETTRA-G2. It is identical to the organization of NETTRA-G1 (Fig. 2.1) except for the insertion of the new subroutine PROCIV. As in Fig. 2.1, an arrow from block i to block j indicates that the subroutine represented by block j is called by the subroutine represented by block i.

3.1 Flowchart of the Subroutines Realizing the Procedure

Although the procedure realized by NETTRA-G2 is quite different from that realized by NETTRA-G1, both of these programs use the subroutine

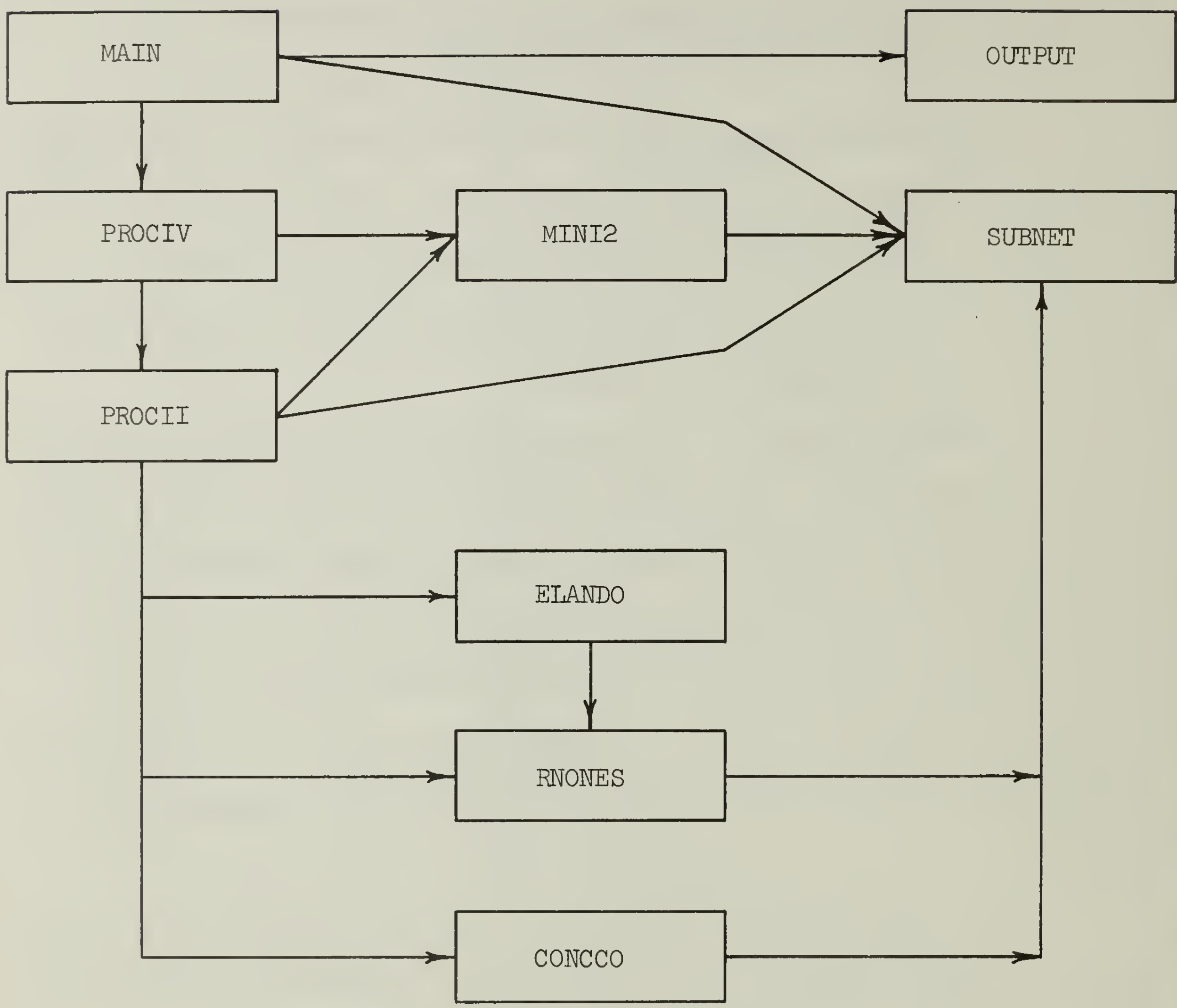


Fig. 3.1 General organization of the program NETTRA-G2.

PROCII to execute the main parts of the transformations. In NETTRA-G2, however, a new subroutine, PROCIV (for PROCedure IV, a name assigned during the development of the program), has been added to control the application of PROCII to the network.

PROCIV is a very simple subroutine. Its flowchart is shown in Fig. 3.1.1.

Block 1 of Fig. 3.1.1 calls MINI2 (a subroutine described in detail

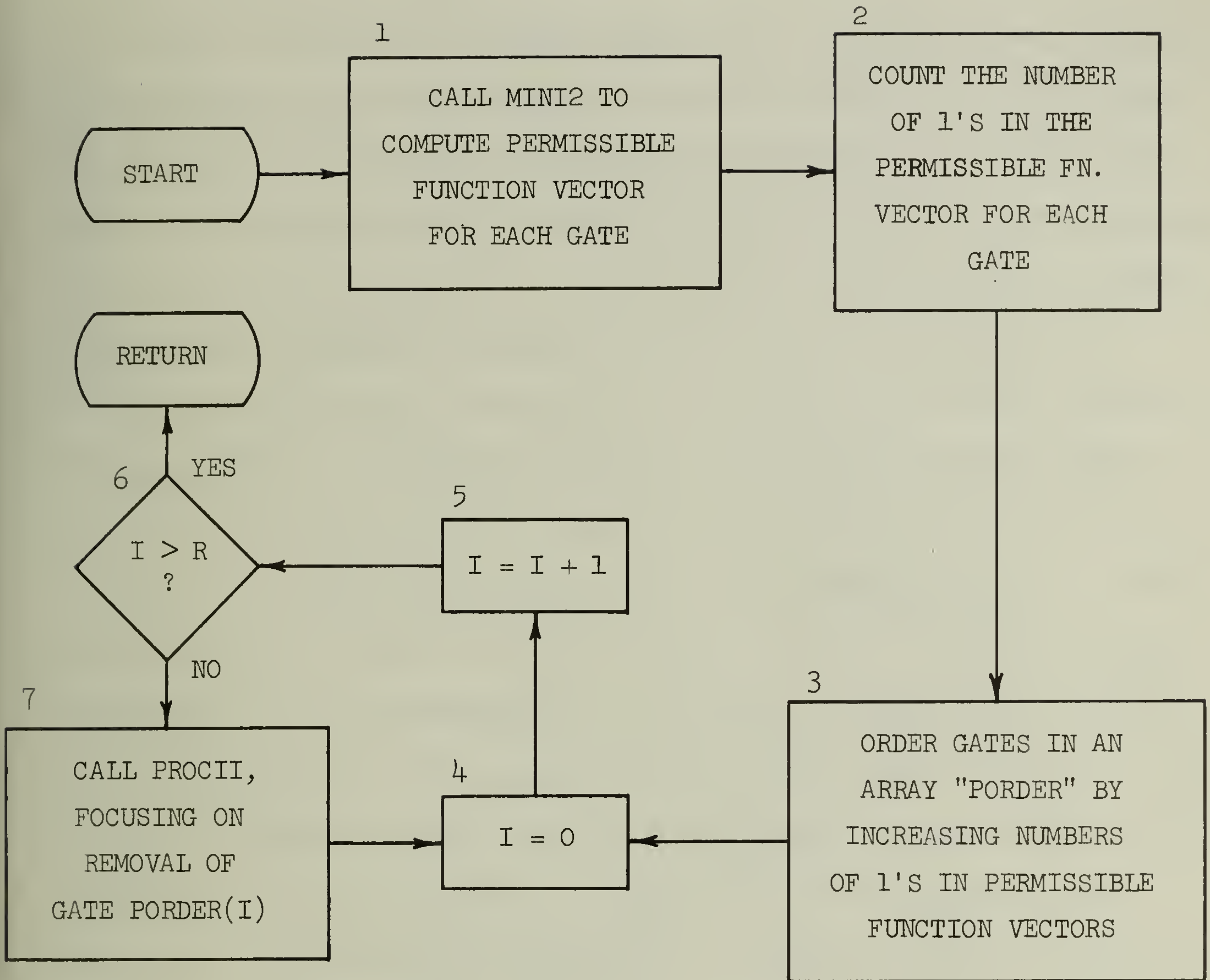


Fig. 3.1.1 Flowchart of PROCIV.

in [3]) to compute a permissible function vector for each gate. During this process some gates may be removed, but this probably will not happen unless the network contains a sufficient amount of redundancy. In any case, the purpose of calling MINI2 at this point is mainly to determine a set of permissible function vectors. Block 2 then counts the number of 1's appearing in the permissible function vector of each gate. This

information is used in block 3 to make an ordering of gates based on increasing numbers of 1's in the respective permissible function vectors of the gates. The ordered gate labels are stored in the array PORDER (such that PORDER(1) is a gate which has the least number of 1's in its permissible function vector, and PORDER(R) is a gate which has the greatest number).

Blocks 4, 5, 6, and 7 form a loop which repeatedly calls a special version of PROCII (the subroutine's characteristics are modified by specifying a certain parameter during the call to that subroutine). PORDER(1), PORDER(2), ... PORDER(R) are attempted to be removed from the network by the special PROCII - one gate, PORDER(I), upon each loop through block 7.

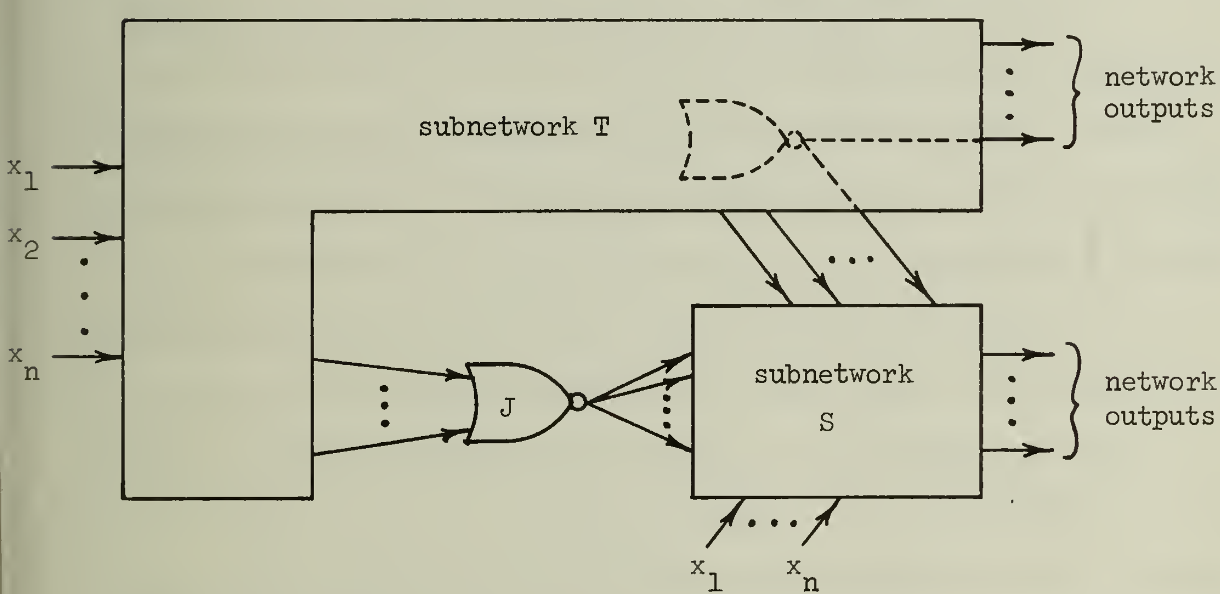
Each time PROCII is called specifying a particular gate, PORDER(I), the subroutine attempts the removal of that specific gate. The meaning of this will become clearer during the discussion of the flowchart of the "modified" PROCII. The reason for creating the ordering PORDER is to try to remove the more easily removable gates (gates with more 1's in their permissible function vectors are, in general, more essential to the network and thus are more difficult to remove) first, before they become more deeply entangled (e.g., by using their outputs as new connections to enable the removal of some other gate(s)) in the network.

In the following discussion of the modified PROCII (let it be referred to as PROCII-M to distinguish it from the regular PROCII) a general knowledge of the information contained in [1] will once again be assumed.

Fig. 3.1.2 presents a diagram of a general multiple-output network [note the possibility that some of the network outputs of subnetwork T may be used as inputs to S (see gate drawn in dotted lines)] to be

transformed by PROCII-M in an attempt to remove the gate labeled "J". (Arrows have been added only to indicate the direction of signal propagation through the connection lines.) Notice that all of the gates on every path from gate J to an output gate are contained in subnetwork S; every other gate in the network, besides J itself, is in subnetwork T. This distinction of whether each gate belongs to T or S is also made by the program itself when PROCII is modified to PROCII-M.

Fig. 3.1.2 Generalized network for the application of PROCII-M.



Generally PROCII-M operates in much the same way as PROCII, except: (1) while gates in T (and external variables) may be connected to gates in S (or T) during the transformation, gates in S are not permitted to be

connected to any other gates; (2) covering of zeroes in gates of subnetwork T is not considered;[†] (3) whenever there is a choice between the removal of a non-essential connection from a gate in T (or from an external variable) and one from a gate in S, the connection from the gate in S is removed; and (4) during the assignment of covering 1's (e.g., during block 23 of Fig. 2.1.2) external variables and gates in T are preferred covers. These modifications tend to push some of the burden of logic from subnetwork S to subnetwork T. In turn, this tends to reduce the necessity of gate J - hopefully until it is no longer needed at all. If this point can be reached, gate J may be removed from the network.

The above 4 differences are achieved by the following program modifications[‡] respectively:

{1} In block 11 of the flowchart of PROCII (see Fig. 2.1.2) only external variables or the outputs of gates in T are allowed to become connectable functions.

{2} In block 8 (see Fig. 2.1.1), if GCO is found to be a gate in subnetwork T, control of the program is sent back to block 3. If GCO is not in T, the regular tests (to see if GCO is an external variable or isolated gate) are then made.

{3} Blocks 10 and 18 (Fig. 2.1.2) are the only parts of the program which eliminate non-essential inputs to each GCO. This is done in both blocks by calling the subroutine ELANDO. So difference (3) is effected by having

---

† By specifying a certain parameter, covering of zeroes for gates in T can also be effected.

‡ As previously mentioned, these modifications are built into the program and are trigered by the specification of the appropriate parameter during the call of PROCII . No reprogramming is required.

ELANDO first remove non-essential connections from gates in S to GCO. If any non-essential connections remain, they are then removed in the usual order (as in the normal PROCII).

(4) Block 23 (Fig. 2.1.2) makes the covering assignments. In choosing covers for zeros of GCO, block 23 prefers covers in the following general order:[†] (a) external variables; (b) gates in subnetwork T; (c) gates in subnetwork S. Obviously, covers of type (a) or (b) place no requirements on the gates of subnetwork S.

This ordering, calculated in block 1 of Fig. 2.1.1 just after the determination of GORDER, is stored in the array RORDER. While the ordering of GORDER is still used in blocks 1 and 7 of the flowchart (Fig. 2.1.1), the ordering of RORDER replaces that of GORDER in making the covering assignments in block 23 (in Fig. 2.1.2). The ordering in this RORDER is divided into 3 groups of ordered gates (and external variables). The first group, stored in memory locations RORDER(1), ..., RORDER(k), is an ordering of all of the gates in T and all of the external variables (actually, the external variables are ignored in the use of this ordering although they do appear). The second group contains an ordering of all the gates in S. This group can be found in locations RORDER(k+1), ..., RORDER(NR-1). Finally, the third group consists of only a single gate, J. It is stored in RORDER(NR). The overall preference of gates decreases from RORDER(1) through RORDER(NR).

After covers consisting of external variables have been assigned wherever possible, this preference ordering is employed to select covers for the remaining uncovered 0 components of GCO's permissible function vector.

[†] Actually there are several different orderings for assigning covers which are available to the user. Only one of these will be described in detail here.

If it is desired by the user, he has the option of also carrying out the usual calculations (i.e., calculation of permissible function vectors, rearrangement of inputs, removal of redundant connections, etc.) for the gates of subnetwork T. This is accomplished by specifying a certain parameter while calling PROCII(-M). This parameter blocks the modification {2} discussed above. This might result in the removal of extra connections (and possibly, extra gates), but the necessary computation time is increased. The user should experiment to find whether or not this option is more suitable for solving his class of problems.

3.2. Examples for NETTRA-G2

As mentioned at the end of the previous section, with NETTRA-G2 the user has the option of allowing calculations (permissible function vectors, etc.) to be performed for all gates in the network for every PORDER(I) rather than just for gates in subnetwork S (with respect to the particular PORDER(I)). Examples both with and without this option will be given in this section. Beginning from the same initial network (Fig. 3.2.1), Fig. 3.2.2 and 3.2.3 show the results using NETTRA-G2, respectively, without and with the option.

The initial network, as documented by the printout displayed in Fig. 3.2.1 (a), utilizes 26 gates and 104 connections to produce a single 5-variable output function. Only uncomplemented variables are available as inputs to the network.

The output from NETTRA-G2 next shows the complete truth table for all of the gates of the original network (Fig. 3.2.1 (b)). This is followed by a description of the network's configuration (Fig. 3.2.1 (c)). Every

Fig. 3.2.1 Initial Printout Describing a Network to be Transformed by NETTRA-G2.

```
*******   5 VARIABLE, 1 OUTPUT TEST NETWORK

          NUMBER OF VARIABLES =   5
          NUMBER OF FUNCTIONS =   1
          COST COEFFICIENT A  = 100
                           B  =   1

          --- UNCOMPLEMENTED VARIABLES  X ---


FUNCTION NO.  1.
   10000101100011101100000111001011


ORIGINAL NETWORK     COST= 26104
```

(a) Heading information and network parameters.

TRUTH TABLE

X1	=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2	=	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3	=	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4	=	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5	=	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	=	1	0	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	1	1
2	=	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	=	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	=	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	=	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	=	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	=	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	=	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	=	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	=	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	=	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	=	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	=	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	=	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	=	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
16	=	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	=	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
18	=	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
20	=	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
21	=	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
22	=	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
23	=	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
24	=	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
25	=	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
26	=	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

(b) Truth table for original network.

GATE ..	LEVEL	FED BY
1	/ 1/	3 4 5 6 8 9 13 15 17 19 20 22 24 26
2	/ 3/	X1 X2 X3 X4 X5
3	/ 2/	X1 X2 X3 X4 2
4	/ 2/	X1 X2 X3 2
5	/ 2/	X1 X2 X4 X5 2
6	/ 2/	X1 X2 X5 2
7	/ 3/	X1 X3 X4 X5
8	/ 2/	X1 X3 X4 7
9	/ 2/	X1 X3 X5 2 7
10	/ 3/	X1 X2
11	/ 3/	X1 X4
12	/ 3/	X1 X5
13	/ 2/	X1 10 11 12
14	/ 3/	X2 X3 X4
15	/ 2/	X2 X3 14
16	/ 3/	X2 X3 X4 X5
17	/ 2/	X2 X4 X5 16
18	/ 3/	X1 X2 X4
19	/ 2/	X2 X4 14 16 18
20	/ 2/	X2 X5 2 16
21	/ 3/	X3 X4 X5
22	/ 2/	X3 X5 21
23	/ 3/	X3 X4
24	/ 2/	X3 23
25	/ 3/	X4 X5
26	/ 2/	X4 11 14 18 23 25

(c) Configuration of original network.

Fig. 3.2.2 Transformed Network Obtained by NETTRA-G2 Without Option (see text).

```
NETWORK DERIVED BY PROCIV
TIME ELAPSED=       217   CENTISECONDS

TRUTH TABLE

X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1 1
 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
 8 = 0 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
12 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
13 = 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
14 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
15 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16 = 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
17 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
18 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
20 = 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0
21 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
22 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
23 = 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0
24 = 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0
25 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
26 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0
```

(a) Truth table for transformed network.

GATE .. LEVEL		FED BY
1	/ 1/	8 13 20 24 26
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 3/	X5
8	/ 2/	X1 X3 7
9	/ 1/	
10	/ 3/	X2
11	/ 3/	X1
12	/ 1/	
13	/ 2/	X1 7 10 23
14	/ 1/	
15	/ 1/	
16	/ 3/	X3 X4
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 2/	X2 X5 16
21	/ 1/	
22	/ 1/	
23	/ 3/	X4
24	/ 2/	X3 16
25	/ 1/	
26	/ 2/	X4 7 11 16

* A NETWORK DERIVED BY PROCIV
COST= 11027

(b) Configuration of transformed network.

Fig. 3.2.3 Transformed Network Obtained by NETTRA-G2 With Option (see text).

NETWORK DERIVED BY PROCIV
TIME ELAPSED = 320 CENTISECONDS

TRUTH TABLE

X1 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2 =	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3 =	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4 =	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5 =	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 =	1	0	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	1	1	0	0	1	0	1	1
2 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7 =	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
8 =	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10 =	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
11 =	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
13 =	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
15 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
16 =	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0
17 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
18 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
19 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20 =	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0
21 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24 =	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0
25 =	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0

(a) Truth table for transformed network.

GATE ..	LEVEL	FED BY
1	/ 1/	8 13 20 24 26
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 3/	X5
8	/ 2/	X1 X3 7
9	/ 1/	
10	/ 3/	X2
11	/ 3/	X1 X4
12	/ 1/	
13	/ 2/	X1 7 10 11
14	/ 1/	
15	/ 1/	
16	/ 3/	X3 X4
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 2/	X2 X5 16
21	/ 1/	
22	/ 1/	
23	/ 1/	
24	/ 2/	X3 16
25	/ 1/	
26	/ 2/	X4 7 11 16

* A NETWORK DERIVED BY PROCIV
COST= 10027.

(b) Configuration of Transformed Network.

gate is listed accompanied by a list of the gates and/or external variables which are its inputs.

If NETTRA-G2 is used without the option just discussed (i.e., when PROCIV calls PROCII for gate PORDER(I), calculations are carried out only for GCO $\in$ S(PORDER(I))), the final result would be as shown in Fig. 3.2.2. The truth table for the transformed network appears in Fig. 3.2.2(a) (remember that outputs for disconnected gates are shown as all 1's). The required computation time for the transformation, 2.17 seconds, is also recorded. Fig. 3.2.2(b) contains a description of the corresponding network consisting of 11 gates and 27 connections.

However, if NETTRA-G2 is used with the option (i.e., when PROCIV calls PROCII for gate PORDER(I), calculations are carried out for every gate GCO in the network), the printed results would appear as in Fig. 3.2.3. Again, Fig. 3.2.3(a) shows the truth table corresponding.to the transformed network, and Fig. 3.2.3(b) presents a description of the network's configuration. Note that in this case, the computation time has increased to 3.20 seconds while the network size has been further reduced to 10 gates and 27 connections.

4. MAJOR FUNCTIONS OF COMMON SUBROUTINES

The subroutines realizing the 2 procedures presented in Sections 2 and 3 share the support of the following seven subroutines whose principal functions will be discussed in this section: MAIN, CONCCO, ELANDO, MINI2, OUTPUT, RNONES, SUBNET.

Complete program listings of these seven subroutines can be found in the appendix along with the listings of the subroutines realizing the previously described procedures.

The functions of the common subroutines are as follows:

<u>MAIN</u>: This subroutine repeatedly reads in groups of input data which include information about the given networks, e.g., the number of external variables, whether or not the complements of variables are also available as input variables, the number of output functions, the number of NOR gates, the list of connections, and the truth table of the output functions (see Section 5 for details). Using this information, MAIN constructs the incidence matrix, INC$MX, for the network. INC$MX is a two-dimensional array whose arguments represent gates or external variables. An array element INC$MX(A1, A2) ≥ 1 indicates a connection from A1 to A2; an array element INC$MX(A1, A2) ≤ 0 indicates the absence of a connection from A1 to A2. Next, subroutine SUBNET is called to calculate the level of each gate and to make lists of predecessors and successors (i.e., which gates precede which and which gates succeed which). MAIN then prints out the truth table and the constructed incidence matrix of the original network by calling the subroutine OUTPUT. Finally the desired transduction procedure is applied to the network by calling the subroutine(s) realizing the transformation. The configuration

of the transformed network is stored in INC$MX, replacing the original network. Then MAIN prints the results of the transduction procedure, i.e., the elapsed time for the transformation, the new incidence matrix, and the new truth table.

CONCCO: During the execution of subroutine PROCII (which is involved in both of the transduction procedures described in this manual) sometimes a gate P (or a gate EFFCON) must be connected to a gate GCO (see blocks 15 and 21 of Fig. 2.1.2). Calling CONCCO (GCO, CCO) connects gate CCO to gate GCO and updates most of the arrays which are involved. Subroutine CONCCO has a second entry point, CNCCO. Calling CNCCO (GCO, CCO) has the same effect as calling CONCCO (GCO, CCO), except the call to CNCCO bypasses the update of the array INPTCV.

ELANDO: This subroutine, when called specifying the necessary argument GATE (e.g., CALL ELANDO(GATE)), eliminates non-essential (redundant) input connections to gate GATE and orders the remaining connections in a list L according to decreasing numbers of essential 1's. This process, used in blocks 10 and 18 of the flowchart for PROCII, was discussed in detail in Section 2.1. If all of the input connections to a gate GATE are already known to be essential or if the non-essential connections are not to be removed, a call to ELANDO's entry point, ORDERL(GATE), will perform just the ordering function, making no attempt to search for and remove non-essential inputs to GATE (as in block 22 of PROCII, Fig. 2.1.2).

MINI2: MINI2 is a subroutine which realizes a pruning procedure (i.e., it transforms a network strictly by removing connections). In NETTRA-G1 and NETTRA-G2, it is used as a fast "clean-up" procedure to remove some of the redundant connections remaining after the main transformations have finished. MINI2 is described in some detail in [3].

OUTPUT: This subroutine may be entered at five different points by a call to either OUTPUT, PAGE, LINE, TRUTH, or CKT.

OUTPUT	assigns mnemonic names to external variables and gates for the purpose of achieving a readable print-out.
PAGE	ejects one page on the printer.
LINE	skips a specified number of lines on the print-out sheet. The number is specified by the argument in the call (e.g., "CALL LINE(5)" skips 5 lines).
TRUTH	prints out the truth table of the network currently stored in INC$MX.
CKT	prints out the network configuration.

RNONES: Whenever an input TH is removed from a gate GCO, much work must be done to update the several arrays and variables involved. The bulk of this updating is performed by a call to this subroutine, CALL RNONES (GCO, TH, FLAG). The parameter FLAG (= 0, 1, or 2) is necessary since this subroutine is called from several different points in the program whose updating requirements differ slightly. The value of FLAG during the call to RNONES determines which particular variables and arrays will be updated.

<u>SUBNET</u>: This subroutine may be entered at three different points by a call to either SUBNET, UNNECE, or PVALUE.

SUBNET generates detailed information on the topology of the network stored in INC$MX: (1) It calculates the level of each gate in the network. Level 1 is assigned to gates having no output connections (thus all gates which have been removed from the network will be assigned level 1). (2) It lists all immediate successors and immediate predecessors for each gate. (3) It calculates the successor matrix which is stored in a two-dimensional array, SUC$MX. The value of SUC$MX(A1, A2) indicates the existence or non-existence of a path from gate (or exteral variable) A1 to gate A2.

UNNECE disconnects certain types of obviously unnecessary connections in the network and updates the above information (discussed in (1), (2), and (3)). The connections removed from the given network are those existing in no paths from the external variables to the output gates.

PVALUE calculates the actual truth table for the entire network stored in INC$MX.

5. INPUT DATA SETUP

In order to fully understand the description of the setup of the input data cards, certain preliminary explanations are necessary.

The purpose of network transductions is to reduce the cost of a network which realizes a certain function (or functions) or to alter the network in such a way as to allow another transduction to eventually accomplish such a reduction. This cost, C, is formally defined by the weighted sum of the number of gates, R, and the number of connections†, I, of a particular network, i.e.,

$$C = A \times R + B \times I$$

where weights A and B are arbitrary non-negative numbers.

Suppose the original network which is to be transformed produces m output functions of n variables. Let x_ℓ, $\ell = 1, \ldots, n$, be the external variables and f_h, $h = 1, \ldots, m$, be the output functions. Before a transformation can be performed on a network by a program, a description of that network must be input to the program. In the case when all of the output functions are completely specified (i.e., no "don't cares"), specifying only the interconnection pattern of the network is sufficient. But if one or more of the output functions is not completely specified, then the user must also provide the truth table (truth tables for all output functions are condensed into a single table) of the problem. Providing the truth table to the program consists of two steps,

---

† A "connection" refers to either a connection from an external variable or an interconnection between two gates.

namely the specification of external variables, and the specification of output functions.

The method of specifying the output functions depends directly upon the method chosen to specify the external variables. External variables may be specified in either of two ways, (a) an implicit specification of external variables, or (b) an explicit specification of external variables.

(a) In the case of implicit specification of external variables, the user specifies the number n of external variables along with a parameter which indicates whether or not the uncomplemented variables are available. Reading the value n along with the parameter, the program internally generates the entries of external variables of an ordinary truth table, that is, a truth table which consists of 2^n input vectors, as shown in Fig. 5.1. In this truth table, the input vectors are arranged according to the order such that an integer j expressed in a binary representation $(x_1 \ldots x_n)$ increases, where x_1 is the most significant digit and x_n is the least significant digit. For example, the truth table for a function of three variables is shown in Fig. 5.2.

The implicit specification of external variables is used for logical design problems in which the output functions have relatively few don't-care terms.

The uncomplemented variables	x_1	$x_1^0 \ldots x_1^j \ldots x_1^{2^n-1}$
	$\cdot$	$\cdot$
	$\cdot$	$\cdot$
	$\cdot$	$\cdot$
	x_n	$x_n^0 \ldots x_n^j \ldots x_n^{2^n-1}$
The complemented variables	$\bar{x}_1$	$\bar{x}_1^0 \ldots \bar{x}_1^j \ldots \bar{x}_1^{2^n-1}$
	$\cdot$	$\cdot$
	$\cdot$	$\cdot$
	$\cdot$	$\cdot$
	$\bar{x}_n$	$\bar{x}_n^0 \ldots \bar{x}_n^j \ldots \bar{x}_1^{2^n-1}$
The output functions	f_1	$f_1^0 \ldots f_1^j \ldots f_1^{2^n-1}$
	f_2	
	$\cdot$	$\cdot$
		$\cdot$
	$\cdot$	$\cdot$
	f_m	$f_m^0 \ldots f_m^j \ldots f_m^{2^n-1}$

These entries exist only in the case of logical design problems where the complemented variables are available as external inputs.

Fig. 5.1 The truth table of output functions of n variables

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
$\bar{x}_1$	1	1	1	1	0	0	0	0
$\bar{x}_2$	1	1	0	0	1	1	0	0
$\bar{x}_3$	1	0	1	0	1	0	1	0
f_1	$f_1^0 \ldots\ldots f_1^7$							

These entries exist only in the case of logical design problems where complemented variables are available as input variables.

Fig. 5.2 The truth table of a function of three variables.

(b) In the case of explicit specification of external variables, the user specifies the entries of external variables of the truth table using additional cards called < external-variable-card > s. The explicit specification of external variables is used in the case of logical design problems where output functions have many don't-care terms. Suppose that the output functions are defined for a subset of input vectors of the entire truth table of Fig. 5.1. Let $\vec{x}^{\,j}$, $j = j_1, j_2, \ldots, j_\mu$ denote these input vectors. The user can 'condense' the truth table of Fig. 5.1 into another table shown in Fig. 5.3.

$$
\begin{array}{r|cccc}
 & \multicolumn{4}{c}{\overbrace{\hspace{12em}}^{\text{only } \mu \text{ input vectors}}} \\
\text{The uncomplemented variables} \left\{ \begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right. & \begin{array}{c} x_1^{j_1} \\ \vdots \\ x_n^{j_1} \end{array} & \begin{array}{c} x_1^{j_2} \\ \\ x_n^{j_2} \end{array} & \begin{array}{c} \cdots \\ \\ \cdots \end{array} & \begin{array}{c} x_1^{j_\mu} \\ \\ x_n^{j_\mu} \end{array} \\
\text{The complemented variables} \left\{ \begin{array}{c} \bar{x}_1 \\ \vdots \\ \bar{x}_n \end{array} \right. & \begin{array}{c} \bar{x}_1^{j_1} \\ \vdots \\ \bar{x}_n^{j_1} \end{array} & \begin{array}{c} \bar{x}_1^{j_2} \\ \\ \bar{x}_n^{j_2} \end{array} & \begin{array}{c} \cdots \\ \\ \cdots \end{array} & \left. \begin{array}{c} \bar{x}_1^{j_\mu} \\ \\ \bar{x}_n^{j_\mu} \end{array} \right\} \begin{array}{l} \text{These entries exist only} \\ \text{in the case of logical} \\ \text{design problems where} \\ \text{the complemented variables} \\ \text{are avaiable as external} \\ \text{inputs.} \end{array} \\
\hline
\begin{array}{c} f_1 \\ \vdots \\ f_m \end{array} & \begin{array}{c} f_1^{j_1} \\ \vdots \\ f_m^{j_1} \end{array} & \begin{array}{c} f_1^{j_2} \\ \\ f_m^{j_2} \end{array} & \begin{array}{c} \cdots \\ \\ \cdots \end{array} & \begin{array}{c} f_1^{j_\mu} \\ \\ f_m^{j_\mu} \end{array}
\end{array}
$$

Fig. 5.3 A 'condensed' truth table having only the input vectors $\vec{x}^{\,j}$, $j = j_1, \ldots, j_\mu$, for which the output functions are defined.

Using < external-variable-card > s, the user can set up internally the table of Fig. 5.3 in place of Fig. 5.1.

5.1 Input Data Card Format

For each separate problem, a set of input data cards must be submitted which consists of the following†:

(i) < heading-card >

(ii) < problem-parameter-card >

(iii) < external-variable-card > s

(iv) < output-function-card > s

omitted for certain cases [applies to (iii) and (iv)]

(v) < connection-description-card > s

Both (i) and (ii) will always consist of only a single card, but (iii), (iv), and (v) may each consist of several cards. Furthermore, types (iii) and (iv) are omitted if all output functions are completely specified, and (iii) need only be prepared in the case of the explicit specification of external variables for the truth table. Following is a description of the formats for each type of input card, (i), (ii), (iii), (iv) and (v):

(i) < Heading-card >

This is the first card of the input deck for a problem. This card may contain any alphanumeric information, in columns 1 ~ 80, which may be used for the identification of the problem, but none of the information on this card will be used in the actual computation. This information will be printed on the first page of the output.

---

† The current implementations of the NETTRA programs accept only heading, problem-parameter, and connection-description cards. Eventually it is hoped that these programs will be modified to accept all of the options described in this section.

(ii) < Problem-parameter-card >

This card specifies the nature of the problem the user wants to solve. There are 7 fields in which to specify the parameters with characters and numerals. These fields are as follows:

Cols. 1 ~ 4: An integer, N, which is right-justified.

This number, N, represents the number of external variables, n, of the output functions. Be sure to punch n (rather than 2n) for N in the case of both complemented and uncomplemented variables available.

Cols. 5 ~ 8: An integer, M, which is right-justified.

This number, M, is the number of output functions, m, to be realized simultaneously. Therefore, of course, M will also be the number of output gates in the network.

Cols. 9 ~ 12: An integer, R, which is right-justified.

This number, R, specifies the number of gates which are included in the network. For various reasons, the user may wish to input networks in which one or more of the gates are "isolated" (i.e., are not connected to any other gates). This is permissible as long as these "isolated" gates are also included in the total number of gates, R.

Cols. 13 ~ 16: An integer, A , which is right-justified.

The number A is the value of the non-negative weight for the number of gates in the cost function. (See Table 5.1.1, 'Typical combinations of values A and B for different network reduction problems'.)

Cols. 17 ~ 20: An integer, B , which is right-justified.

The number B is the value of the non-negative weight for the number of connections in the cost function. (See Table 5.1.1.)

Col. 24: A blank 'b'†, or one of the characters, 'C', 'X', 'Y', 'U' or 'V'.

The 'b' or 'C' parameter represents <u>an implicit specification of both the external variables</u> <u>and</u> <u>an implicit specification of the output functions</u> (in this case, the output functions will be calculated from the connection pattern of the network). The 'X' or 'Y' parameter indicates <u>an implicit specification of external variables only</u>. The 'U' or 'V' parameter indicates <u>an explicit specification of external variables</u>. (See summary of these symbols in Table 5.1.2)

The 'b' or 'X' parameter specifies that only uncomplemented external variables are available for the network. The 'C' or 'Y' parameter specifies that both uncomplemented and complemented variables are available for the network. If the user specifies the 'b', 'X', 'C', or 'Y' parameter, the program sets up the truth table by generating a set of 2^n input vectors $(x_1^j, \ldots, x_n^j)$, for $j = 0, \ldots, 2^n - 1$, in the case of a 'b' or 'X' parameter, or $(x_1^j, \ldots, x_n^j, \bar{x}_1^j, \ldots, \bar{x}_n^j)$ for $j = 0, \ldots, 2^n - 1$, in the case of a 'C' or 'Y' parameter.

The 'b' or 'C' parameters should be used for problems in which the output functions contain no don't-care terms. For such problems, the preparation of the < external-variable-card > s and the < output-function-card > s can be dispensed with since the program can calculate completely all output functions using only a description of the

---

† A 'b' stands for a blank (i.e., no character punched).

Network Reduction Problem	Values of A and B
reducing the number of gates only.	A = 1 and B = 0
reducing the number of gates primarily, then reducing the number of connections secondarily.†	A = 100 and B = 1
reducing the number of connections only.	A = 0 and B = 1
reducing the number of connections primarily, then reducing the number of gates secondarily.	A = 1 and B = 100
reducing the sum of the number of gates and the number of connections.	A = B = 1

Table 5.1.1 Typical combinations of values A and B for different network reduction problems.

† Most of the programs in the NETTRA system are oriented toward this reduction problem, so the user will probably find this combination of A and B the most useful.

	uncomplemented variables only available	both complemented and uncomplemented variables available
implicit specification of external variables and output functions	'b'	'C'
implicit specification of external variables	'X'	'Y'
explicit specification of external variables	'U'	'V'

Table 5.1.2 Possible symbols for column 24 of < problem-parameter-card >.

connection pattern of the network (provided by the <connection-description-card>s).

Similarly, the 'X' or 'Y' parameter implies the use of a complete truth table (i.e., 2^n input vectors for n external variables) inside the program. Since from this information the program can easily generate the truth table entries for the external variables, as just mentioned, the < external-variable-card > s are unnecessary. The m < output-function-card > s, however, must still be prepared.

The 'U' parameter specifies that only uncomplemented external variables are available for the network. The 'V' parameter specifies that both uncomplemented and complemented variables are available for the network. In either case, the 'U' or the 'V' parameter, the user <u>must</u> prepare n < external-variable-card > s and m < output-function-card > s. The program sets up the truth table by reading these < external-variable-card > s and < output-function-card > s.

Cols. 25 ~ 28: An integer, NEPMAX, which is right-justified.

This parameter is omitted for all NETTRA programs except those involving "error-compensation" routines. In the cases where NEPMAX is required, a further discussion of this parameter can be found elsewhere in the manual. The abbreviation NEPMAX is a mnemonic for "maximum number of error positions", and the default is $NEPMAX = 2^{(n-1)}$, where n is the number of external variables.

(iii) < External-variable-card > s

In combination with the 'U' or 'V' parameter in column 24 of the < problem-parameter-card >, the n < external-variable-card > s specify the entries of external variables of the truth table of

Fig. 5.3. Each card contains the binary representation of external variable x_ℓ, i.e., $(x_\ell^{j1}, x_\ell^{j2}, \ldots, x_\ell^{j\mu})$, starting from column 1 of the card. The maximum number of bits in a binary representation is limited to 32. (This means the maximum number of input vectors is 32.) If the actual number of bits is less than 32, then a termination symbol '/' (slash) is put on the right of the right-most bit of the binary representation on the first < external-variable-card >. The remaining columns after the termination symbol '/' in the first card, as well as the same columns in the following cards, may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

In the case of the 'V' parameter, the program generates the binary representations corresponding to complemented variables by taking negations of the entries of the < external-variable-card > s. Therefore the user <u>must not</u> provide < external-variable-card > s representing the complemented variables, $\bar{x}_\ell$.

If one of the parameters 'b', 'C', 'X', or 'Y' appears in column 24 of the < problem-parameter-card >, the user <u>does not</u> prepare < external- variable-card > s.

(iv) < Output-function-card > s

The m < output-function-card > s specify the set of m output functions to be realized simultaneously. Each card contains the binary representation of one output function f_h, starting from column 1 of the card. A symbol '*' is used to denote don't-care terms, if any. The maximum number of bits in a binary representation is limited to 32.

The actual number of bits must be 2^n in the case of an implicit specification of external variables, or must be the same as defined by the < external-variable-card > s in the case of an explicit specification of external variables. The remaining columns, up to column 72 (inclusive), may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

If either the 'b' or 'C' parameter appears in column 24 of the < problem-parameter-card >, the < output-function-card > s must be omitted.

(v). < Connection-description-card > s

In the present version of the program, 9 cards (some of which may be just blank cards) are required.† Each of these cards is divided into 16 fields of 5 columns each (i.e., columns 1 ~ 5, 6 ~ 10, 11 ~ 15, ..., 71 ~ 75, 76 ~ 80). Beginning with the first field of the first card, continuing through the succeeding fields of that card and through the fields of as many additional cards as necessary (up to a maximum of 9, total), the expressions (explained in the next paragraph) C_1, C_2, C_3, ..., are punched right-justified in their respective fields.

Each gate of the network is labeled uniquely by assigning it one of the integers 1, 2, ..., R, such that the output gates receive

---

† For many uses, the user will probably find that these 9 cards far exceed his needs, and may thus be inconvenient. In such a case, the number of required cards may be easily adjusted by making the obvious changes in two statements (A READ statement and a DO statement) following the comment card "C**** READ IN NETWORK INFORMATION AND SET UP INC$MX *****" in subroutine MAIN.

the labels 1, 2, ..., m. The names X1, X2, ..., Xn are assigned to the external variables $x_1, x_2, \ldots, x_n$ (and the names Y1, Y2, ..., Yn to the complemented external variables $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, if appropriate).†
Now, for each connection of the network (i.e., including both the connections from external variables to gates and connections from gates to other gates), a 4 character expression, C_i, is formed, to represent that connection as follows: to represent a connection from gate GI to gate GJ, the numeric label GI is inserted into the first two character positions of C_i and the numeric label GJ is inserted into the second two positions (e.g., the C_i for a connection from gate 9 to gate 5 would be "0905"); to represent a connection from external variable XI to gate GJ, the alphanumeric label XI is inserted into the first two character positions of C_i and the numeric label GJ into the second two positions (e.g., the C_i for a connection from external variable x_3 to gate 10 would be "X310").

Every connection of the network must be represented by a C_i, although there are no restrictions on the order in which the connections (i.e., C_i's) are punched onto the input cards.

---

† At the time of writing, the programs have not yet been changed to recognize this new labeling system. The old labels are simply: 1, 2, ..., n, for external variables $x_1, x_2, \ldots, x_n$ (and $n+1$, $n+2$, ..., $2n$ for the complemented variables $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, if they are permitted in the problem); $n+1$, $n+2$, ..., $n+m$ for the m output gates of the network ($2n+1$, $2n+2$, ..., $2n+m$ if complemented variables are included); and finally $n+m+1$, $n+m+2$, ..., $n+R$ ($2n+m+1$, $2n+m+2$, ..., $2n+R$) for the non-output gates of the network.

These five groups of cards, (i), (ii), (iii), (iv) and (v) in the correct order constitute the necessary description for a single problem. In order to solve several problems during the same computer run, the descriptions for the desired problems are just arranged serially. See Fig. 5.1.1 for an example of the sequencing of all cards for the execution of a NETTRA program using typical JCL statements for the IBM 360/75.

```
// ID  . . .
//      EXEC FORT
// FORT.SYSIN DD*
```

The Program

```
FORTRAN PROGRAM
     SUBROUTINES
```

```
/*

//   EXEC LKED

//   EXEC GO

//GO.SYSIN DD *
```

The Input Data Cards

The first problem

```
< heading-card >
< problem-parameter-card >
< external-variables-card > s
< output-function-card > s
< connection-description-card > s
```

The second problem

```
< heading-card >
        .
        .
        .
```

```
        .
        .
        .
```

The last problem

```
< heading-card >
        .
        .
        .
```

```
/ *
```

Fig. 5.1.1 Input card sequence for the execution of a typical NETTRA program.

5.2 Restrictions on Problem Size

In order to fit the programs into a finite amount of space, some restrictions on the size of an acceptable problem are required:

1. The number t of input vectors in the truth table is 32 or less.
2. The number n of external variables.

   Because of $t \leq 32$, n is 5 or less in the case of completely specified functions. In the case of incompletely specified functions, however, any $n \leq 20$ is acceptable if only uncomplemented variables are available, or $n \leq 10$ if both uncomplemented and complemented variables are available, provided that the truth table is defined by the < external-variable-card > s.
3. The number R of gates.

   The number of gates, R, may not exceed 40-n in the case of only uncomplemented variables available (a 'b', 'X', or 'U' parameter). In the case of both uncomplemented and complemented variables available (a 'C', 'Y' or 'V' parameter), the limit is lowered to 40-2n.

All of these limitations are essentially imposed by the array sizes in the programs as presently written. To loosen the restrictions is mainly a task of increasing array dimensions appropriately.

5.3 Examples of Input Data Setup

The following examples will illustrate, for the general program in the NETTRA system, various possible input data card setups complying with the directions given in Section 5.1.

Example 1: A two output network of four variables shown in Fig. 5.3.1. Assume the two output functions are $f_1 = CCEF$† and $f_2 = 3BBB$ and only uncomplemented variables are available. Furthermore, assume it is desired to reduce the number of gates primarily and the number of connections secondarily (see Table 5.1.1).††

From this description, the < problem-parameter-card > must contain the following values:

Cols. 1 ~ 4	4,	the number of external variables
Cols. 5 ~ 8	2,	the number of output functions
Cols. 9 ~ 12	8,	the number of gates in the original network
Cols. 13 ~ 16	100,	the value of A
Cols. 17 ~ 20	1,	the value of B
Cols. 24	'b',	uncomplemented variables only available and implicit specification of both the external variables and the output functions
Cols. 25 ~ 28	'b',	since the NEPMAX parameter is unrelated to the program to be used

Fig. 5.3.2 shows the setup of data cards used to specify the network in Fig. 5.3.1 as input for the program. Notice that in forming the C_i, the four uncomplemented variables are represented by the labels X1, X2, X3, X4; the two output gates by the numbers 1, 2; and the remaining gates, by the numbers 3, 4, 5, 6, 7, 8. This manner of labeling is

† For convenience, the output functions are expressed in hexidecimal notation. When the numbers in this notation are expanded into binary, they represent the output vectors as they appear (i.e., in the same left-to-right order) in the complete truth table described earlier and pictured in Fig. 5.1.

†† This assumption is implicit in most of the transduction procedures and their implementations which comprise the NETTRA system.

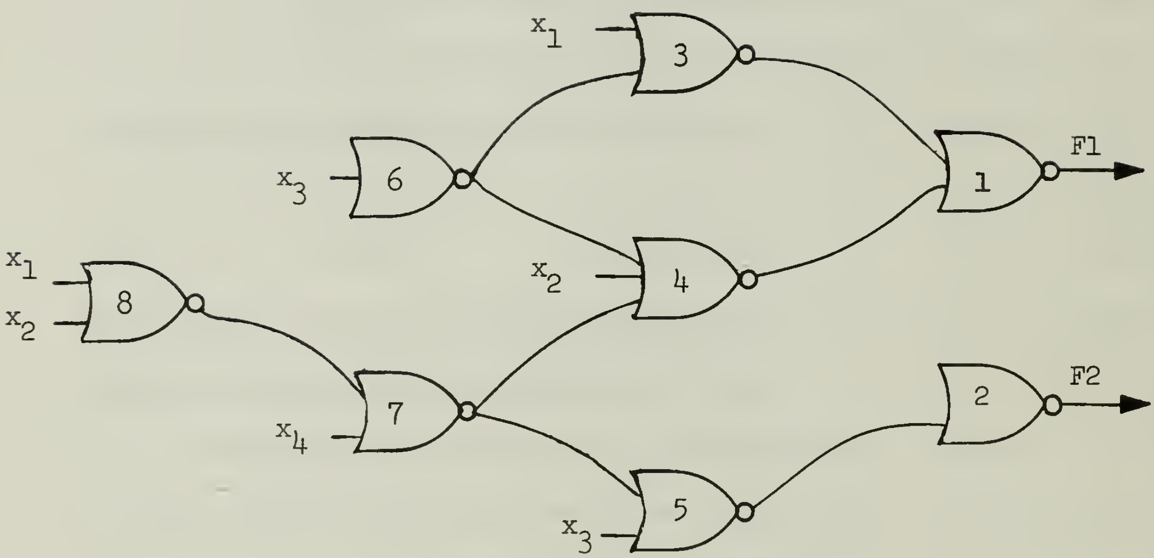


Fig. 5.3.1 Network to be transformed in Examples 1 and 2.

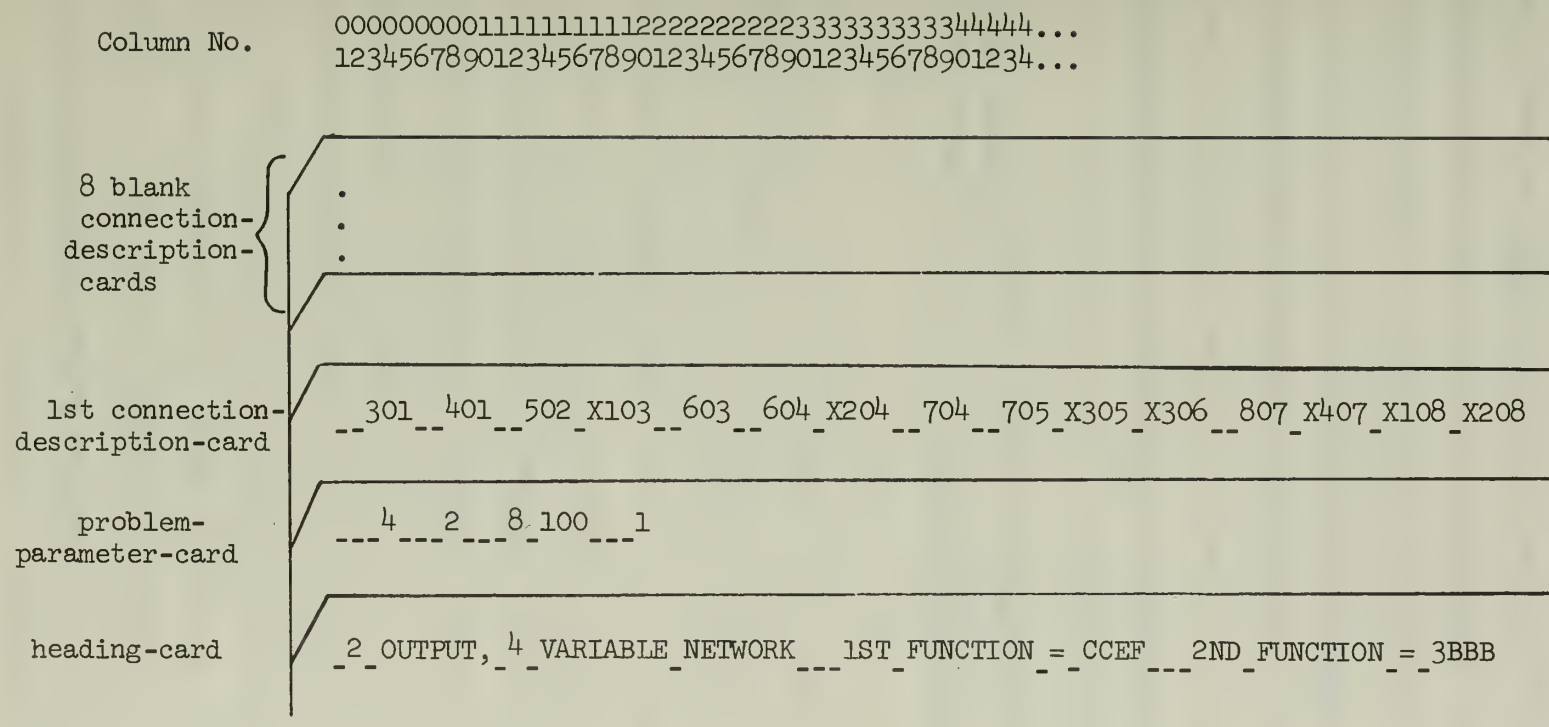


Fig. 5.3.2 Possible setup of data cards to specify the problem given in Example 1.

strictly required by the instructions for preparing the < connection-description-card > s (see Section 5.1).

The heading card in Fig. 5.3.2 will simply be read by the program and printed character for character onto the output page as an identification of the particular problem. Below that, the number of variables, number of functions, and the cost coefficients, A and B, will be printed (all with appropriate labels). Also, immediately following will be a statement of what types of external variables are permitted (i.e., either just uncomplemented variables or both complemented and uncomplemented) along with their generic names:

$$\left.\begin{array}{l}\text{X - for uncomplemented variables}\\ \text{Y - for complemented variables}\end{array}\right\}\quad\begin{array}{l}\text{if external variables}\\ \text{were implicitly specified}\end{array}$$

<u>or</u>

$$\left.\begin{array}{l}\text{U - for uncomplemented variables}\\ \text{V - for complemented variables}\end{array}\right\}\quad\begin{array}{l}\text{if external variables}\\ \text{were explicitly specified}\end{array}$$

For example, if both X and Y appear as generic names (as would occur in the case of an implicit specification of external variables with both complemented and uncomplemented variables available) then the external variable names which appear on subsequent output pages will be X1, X2, ..., Xn and Y1, Y2, ..., Yn. Or, if both U and V appear as generic names (as would occur in the case of an explicit specification of external variables with both complemented and uncomplemented variables available) the external variable names which appear in the output will be U1, U2, ..., Un (for the uncomplemented variables) and V1, V2, ..., Vn (for the complemented variables). It should be noted, however, that the letters U and V, as used as replacements for X and Y (respectively) in the

naming of external variables (e.g. U1, V1 instead of X1, Y1), appear strictly on the output pages of the program - they are not used internally in the program and they must not appear in the variable names punched on the < connection-description-card > s by the user. They are intended only as an aid to the user so that, at a glance at the transformed network in the output, he can easily distinguish whether the external variables were implicitly or explicitly specified for that particular problem.

Following the statement of whether only uncomplemented or both complemented and uncomplemented external variables are employed, the user will find next on the output page the cost of the original network which was input to the program. This is the cost which was defined in the beginning of Section 5.

The cost will be followed by a truth table (generally in the same form as Fig. 5.1) showing the outputs (0 or 1) of all of the gates in the network for every external variable input combination (i.e., combinations of 0's and 1's) of interest.

Finally, below the truth table will be printed a description of the network submitted as input. This is for documentation purposes, and it is also much more readable than the network description which appeared on the < connection-description-card > s. In this description, each gate is listed along with the names of the gates and external variables which feed it. Also, to assist the user in sketching the network from its description, the level of each gate in the network is included (gates which do not feed other gates are assigned to level 1, all other gates are assigned level numbers such that each gate is in a level one

higher than the highest level gate directly fed by it).

All of the information just described will be printed before the execution of the transduction actually begins. This will be followed, beginning at the top of a new output page, by the network(s) actually obtained as a result of the computation. First the complete truth table of the transformed network will be printed, followed by a network connection description of the form just described above. Finally, the cost of the new network will be calculated and printed.

In this example, it was assumed that there were no "don't-cares" in the output functions implicitly specified by the input, thus no < external-variable-card > s or < output-function-card > s were included. In the next example, however, < output-function-card > s will be required in order to specify some of the components of the output functions as "don't-cares".

<u>Example 2</u>: The two output network of four variables shown in Fig. 5.3.1. This is the same network used in Example 1, but this time the output functions are not assumed to be completely specified. Let f_1 = '11001**01*10*111' and f_2 = '0**110111*111011' be the required functions. Also, suppose that both complemented and uncomplemented variables are desired to be available during the transduction. Again the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this problem, the following values must appear on the < problem-parameter-card >:

Cols.	1 ~ 4	4, the number of external variables
Cols.	5 ~ 8	2, the number of output functions

Cols. $9 \sim 12$	8,	the number of gates in the original network
Cols. $13 \sim 16$	100,	the value of A
Cols. $17 \sim 20$	1,	the value of B
Col. 24	Y,	indicative of an implicit specification of external variables and the availability of both complemented and uncomplemented variables

Fig. 5.3.3 shows the setup of the data cards corresponding to this problem. Notice the differences and similarities to the data cards shown in Fig. 5.3.2. The < problem-parameter-card > differs only in column 24. The < external-variable-card > s are missing in both Fig. 5.3.2 and Fig. 5.3.3 since the external variables are implicitly specified for both problems. The < output- function-card > s, however, appear in Fig. 5.3.3 but not in 5.3.2 since they are necessary to specify "don't-care" components which do not occur in the completely specified output functions of Example 1. In both cases, though, the < connection-description-card > s are identical since the original networks are identical.

By allowing "don't-care" terms in the output functions, and by allowing the use of both complemented and uncomplemented variables† (even though the original network employed only uncomplemented variables), the restrictions during the transduction process are loosened (compared to what they were for Example 1), perhaps permitting a network of

---

† In the case of NETTRA-PG1, -P1, and -P2, it is useless to specify Y rather than X in column 24 for this example. Since the original network uses only uncomplemented variables, to these programs which perform "pruning" procedures (i.e., procedures which are incapable of adding new connections) the availability of complemented variable is not meaningful.

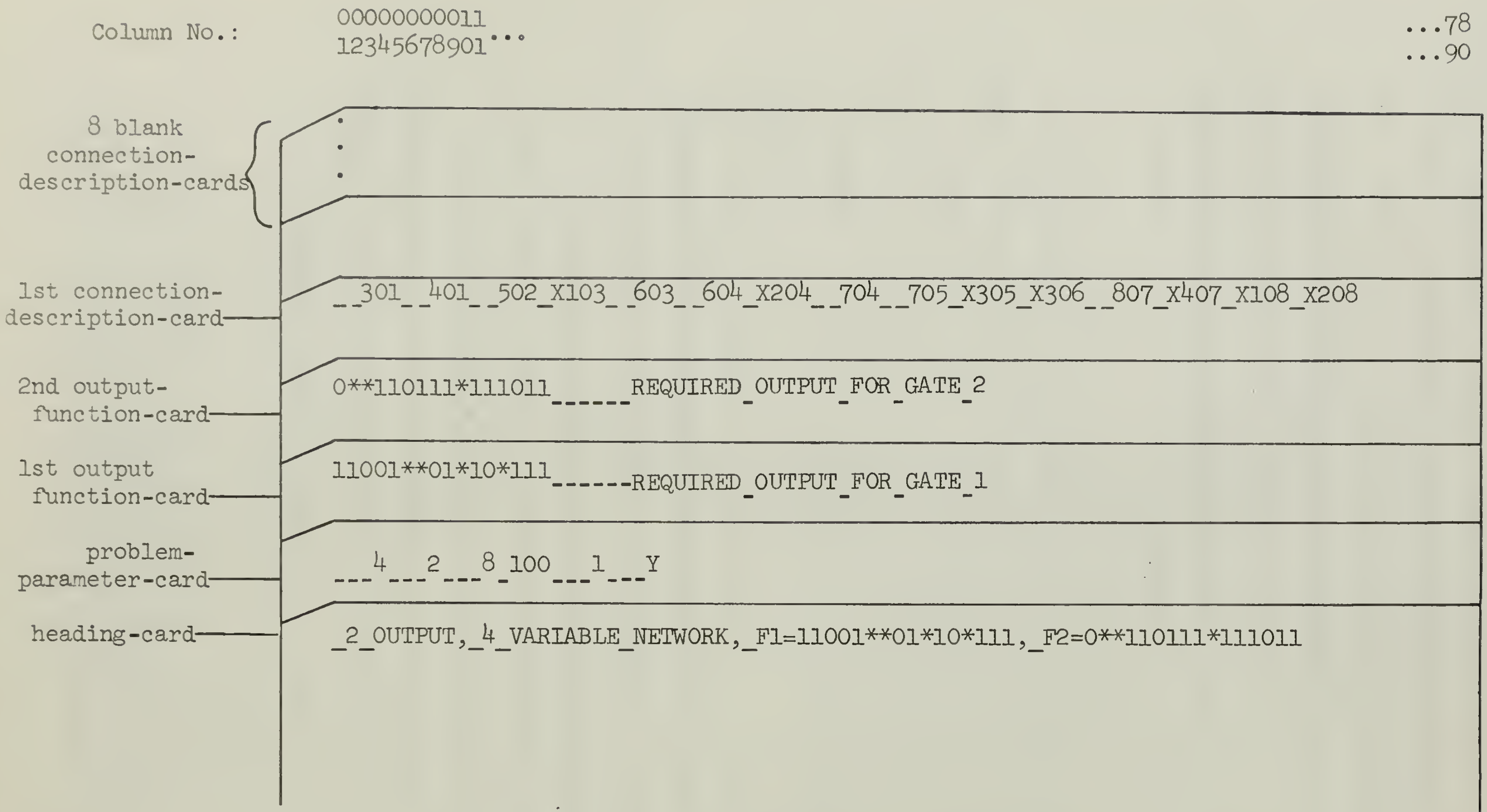


Fig. 5.3.3 Possible setup of data cards to specify the problem given in Example 2.

less cost to be obtained.

Notice that the first < output-function-card > corresponds to the output of gate 1 and the second < output-function-card > corresponds to the output of gate 2. This must hold true for every problem in which < output-function-card > s are included; the gates labeled 1, 2, ..., m must correspond to the output functions specified on < output-function-card > s 1, 2, ..., m, respectively.

Of course, the printed output of the program will be in the same format described in Example 1.

Example 3: The three output network of six variables shown in Fig. 5.3.4. The outputs are again assumed to be incompletely specified. In fact, only the following 11 input combinations are specified out of a possible 64 ($=2^6$):

x_1	0	0	0	0	0	0	0	0	0	0	1
x_2	0	0	0	0	0	0	0	1	1	1	0
x_3	0	0	0	0	0	0	0	0	0	1	1
x_4	0	0	0	0	1	1	1	0	0	0	1
x_5	0	0	1	1	0	0	1	0	1	1	0
x_6	0	1	0	1	0	1	0	1	1	0	0
F_1	0	0	1	1	0	0	*	0	0	0	0
F_2	1	1	*	1	1	1	0	1	1	0	*
F_3	1	1	0	0	0	0	0	1	0	0	0

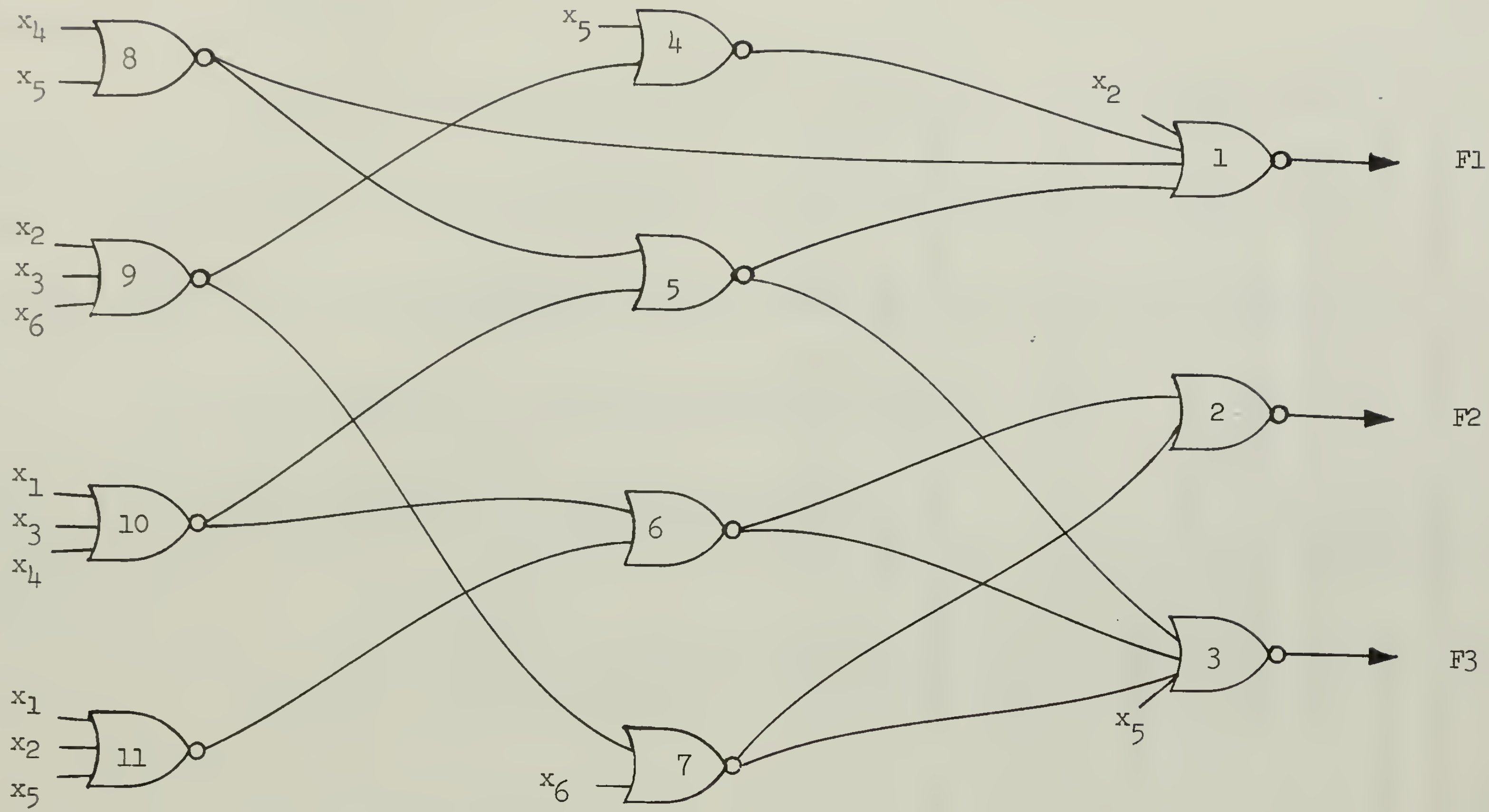


Fig. 5.3.4 Network to be transformed in Example 3.

Additionally, only uncomplemented variables are assumed to be available, and the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this example, the following parameters appear on the < problem-parameter-card >:

Cols.	1 ~ 4	6,	the number of external variables
Cols.	5 ~ 8	3,	the number of output functions
Cols.	9 ~ 12	11,	the number of gates in the original network
Cols.	13 ~ 16	100,	the value of A
Cols.	17 ~ 20	1,	the value of B
Col.	24	U,	indicative of an explicit specification of external variables and the availability of only uncomplemented variables

Fig. 5.3.5 shows a possible setup of the data cards corresponding to this example. Notice that in this example, the <external-variable-card > s are included, whereas in the two previous examples they were omitted. Although this problem is not too realistic (none of the 3 functions is actually a 6-variable function), it demonstrates the input data preparation to be used in cases where many external variables are present and a high percentage of "don't care" terms exist.

Again, the printed output from the program will follow the same format described in Example 1.

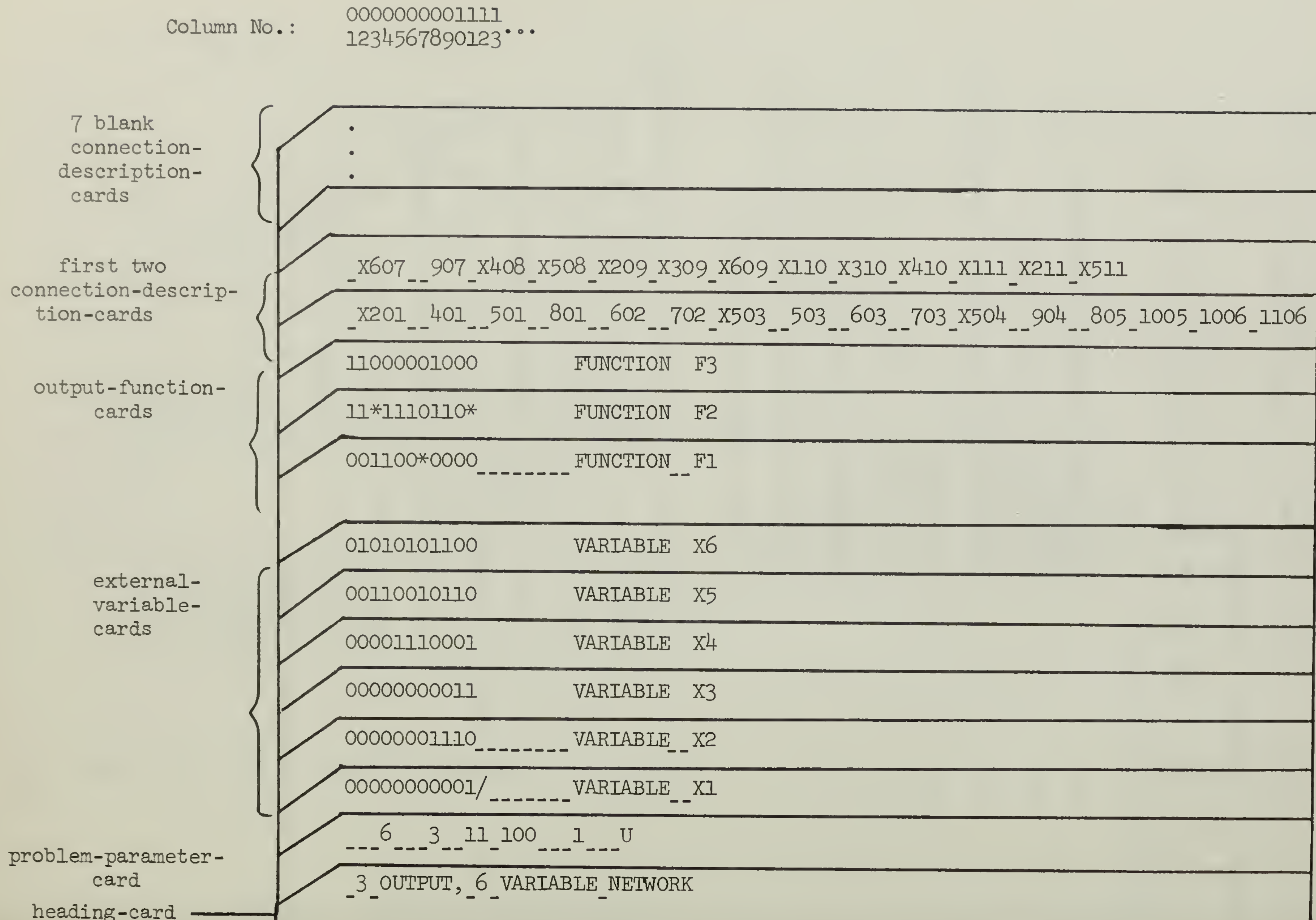


Fig. 5.3.5 Possible setup of data cards to specify the problem given in Example 3.

REFERENCES

[1] Y. Kambayashi and J.N. Culliney, "NOR network transduction procedures based on connectable and disconnectable conditions (Principles of NOR network transduction programs NETTRA-G1 and NETTRA-G2)," to appear as a Report, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.

[2] Y. Kambayashi and S. Muroga, "Network transduction based on permissible functions (General principles of NOR network transduction NETTRA programs)," to appear as a Report, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill.

[3] H.C. Lai and J.N. Culliney, "Program manual: NOR network pruning procedures using permissible functions (Reference manual of NOR network transduction programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)," Report No. UIUCDCS-R-74-686, Dept. of Comp. Sci., Univ. of Ill., Urbana, Ill., Nov. 1974.

APPENDIX:

Program Listings

Following are the listings of the FORTRAN programs NETTRA-G1 and NETTRA-G2. These programs realize, respectively, the transduction procedures discussed in Sections 2 and 3.

Since NETTRA-G2 differs just in two subroutines from NETTRA-G1, only those two subroutines are listed for NETTRA-G2: MAIN (differs only slightly from MAIN in NETTRA-G1) and PROCIV (not contained in NETTRA-G1).

Explanations of variables used in the programs can be found in the listings themselves.

```
C
C
C***************************************************************************************
C                                                                                     *
C                                                                                     *
C          PPPP    RRRR     OOO     GGG    RRRR      A       M      M                 *
C          P   P   R   R   O   O   G   G   R   R    A A      MM    MM                 *
C          P   P   R   R   O   O   G       R   R   A   A     M M  M M                 *
C          PPPP    RRRR    O   O   G  GG   RRRR    AAAAA     M  M   M                 *
C          P       R  R    O   O   G   G   R  R    A   A     M      M                 *
C          P       R   R    OOO     GGG    R   R   A   A     M      M                 *
C                                                                                     *
C                                                                                     *
C                                                                                     *
C N   N   EEEEE   TTTTT   TTTTT   RRRR     A               GGG     1                  *
C NN  N   E         T       T     R   R   A A             G   G   11                  *
C N N N   E         T       T     R   R  A   A            G        1                  *
C N  NN   EEE       T       T     RRRR   AAAAA   XXXXX    G  GG    1                  *
C N   N   E         T       T     R  R   A   A            G   G    1                  *
C N   N   EEEEE     T       T     R   R  A   A             GGG    111                 *
C                                                                                     *
C                                                                                     *
C***************************************************************************************
C
C

      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C         $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C        $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C        $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C              ROW IS USED.
C         $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C           A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C           B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C        COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C       ESS1S: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C              (POSITIONS IN ESS1S CORRES. TO GATES NOT FEEDING GCO ARE
C              IGNORED).
C      F$UB1: POINTS TO LAST ELEMENT IN F$1.
C         F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C              COVERING) IN A CONNECTIBLE FUNCTION.
C          GI: LABEL OF A PARTICULAR GATE.
C      GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C      GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C       HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C              WORK LEVEL J.
C        IDXO: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IDXOE: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C        IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
```

```
C       IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.
C      INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE
C              (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.
C      INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTS
C              HAVE A '1' IN THE POSITION INDICATED BY F$1.
C       IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE
C              TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.
C       IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. IN
C              A LIST OF GATES AND EX. VAR. FEEDING GJ.
C       ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.
C       JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.
C        KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN
C              PERFORMED.
C        KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS
C              BEEN COMPENSATED.
C       KFLAG: SAME AS KEIFLG IN PROCII.
C        LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO
C              ASSIGNED LEVELS JUST LIKE GATES).
C      LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF
C              NETWORK.
C         LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.
C      LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.
C       LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY
C              DECREASING NO. OF 0'S IN GCO COVERED.
C       LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED
C              GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY
C              DECREASING NO. OF ESSENTIAL 1'S.
C      LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR
C              EX. VAR.) GI.
C       LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.
C      LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB
C              CONCERNING GI.
C           M: NUMBER OF NETWORK OUTPUT GATES.
C           N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.
C      NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX
C              ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,
C              PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S
C              REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL
C              TO ONE HALF OF N2 BY DEFAULT.
C          NM: SUM OF N PLUS M
C         NM1: SUM OF NM PLUS 1.
C         NN2: PRODUCT OF N AND N2.
C         NOS: NUMBER OF ELEMENTS IN SET S.
C        NOS1: NUMBER OF ELEMENTS IN SET S1.
C      NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C        NOS2: NUMBER OF ELEMENTS IN SET S2.
C        NOT1: NUMBER OF ELEMENTS IN SET T1.
C      NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C         NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.
C        NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.
C         NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.
C        NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.
C          NR: SUM OF N PLUS R.
C        NRN2: PRODUCT OF NR AND N2.
C       NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)
C                                                          FOR I=1,2.
C          N1: SUM OF N PLUS 1.
```

```
C        N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED    G1 00960
C            (USUALLY 2 TO THE POWER N).                                 G1 00970
C    ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.         G1 00980
C            ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.         G1 00990
C        P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND G1 01000
C            EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),....,    G1 01010
C            P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR   G1 01020
C            SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR   G1 01030
C            FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR        G1 01040
C            CALCULATIONS ASSOCIATED WITH P$(1,-).                      G1 01050
C       PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE        G1 01060
C            REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED    G1 01070
C            NETWORK.                                                   G1 01080
C    POINTA: NOT USED.                                                  G1 01090
C    POINTC: POINTS TO LAST ELEMENT IN LISTC.                           G1 01100
C    POINTL: POINTS TO LAST ELEMENT IN LISTL.                           G1 01110
C    POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).    G1 01120
C     POTAB: POSSIBLE OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL        G1 01130
C            COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY     G1 01140
C            USEFUL) FUNCTIONS.                                         G1 01150
C    PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF   G1 01160
C            ROWS OF POTAB CONCERNING GI.                               G1 01170
C         R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO      G1 01180
C            NOTE SOME OF R GATES MAY BE ISOLATED).                     G1 01190
C      RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE   G1 01200
C                      ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.          G1 01210
C            RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE   G1 01220
C                      AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.       G1 01230
C    RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).    G1 01240
C    RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).   G1 01250
C         S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST     G1 01260
C            ENTRY IN RSCONN.                                           G1 01270
C      SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATIONG1 01280
C            WHICH ARE TO BE REPLACED IF POSSIBLE.                      G1 01290
C     SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE        G1 01300
C            REPLACED BY ELEMENTS IN SET S2.                            G1 01310
C     SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR    G1 01320
C            REPLACING ELEMENTS IN SET S.                               G1 01330
C     SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS INGI 01340
C                                                             SET S1.   G1 01350
C       STS: STARTING ELEMENT OF SET S.                                 G1 01360
C    SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.   G1 01370
C            SUC$MX(GI,GJ)=0 IF NOT.                                    G1 01380
C      SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.  G1 01390
C     SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.                      G1 01400
C         T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO   G1 01410
C            LAST ENTRY IN RTCONN.                                      G1 01420
C      TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.          G1 01430
C     UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.           G1 01440
C    VF$UB1: POINTS TO LAST ELEMENT IN VF$1.                            G1 01450
C      VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONSG1 01460
C            (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING   G1 01470
C            ORIGINALLY CONNECTED INPUTS TO GCO.                        G1 01480
C                                                                       G1 01490
C                                                                       G1 01500
C                                                                       G1 01510
      COMMON NEPMAX                                                     G1 01520
      COMMON   N            , M            , A            , B           G1 01530
     1    ,    R            , N2           , N1           , NR          G1 01540
     2    ,    NM           , KFLAG        , JFLAG        , COST        G1 01550
     3    ,    LEVM         , NRN2         , NM1          , NN2         G1 01560
```

```
      COMMON    ISUCC(40,40) , LISUCC(40)     , IPRED(40,40) , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)     , UNAME(40)
     2     ,    GLEVEL(40)    , LGLIST(40)     , HLIST(40,40) , TIME
      COMMON    T               , RTCONN(100)    , S            , RSCONN(100)
      COMMON    IFLAG           ,POINTA          ,ESS1S(40)     ,F$1(32)
     1    ,F$UB1              ,INPTCV(32)      ,LISTC(40)     ,POINTC
     2    ,LISTL(40)          ,POINTL          ,ORIGIN(40)    ,IPATH(40)
     3    ,POINTR             ,VF$1(32)        ,VF$UB1        ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)    ,NRPLC(2)
     1    ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)     ,IDX1(32)
     2    ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)     ,NOT1
     3    ,SETS1(40)          ,NOS1            ,SETS(40)      ,NOS
     4    ,STS                ,SUMS2(32)       ,SETS2(200)    ,NOS2
     5    ,LIP                ,NO0E            ,KEYA          ,KEYB
     6    ,NO0                ,NO1             ,NO1E          ,$GT
     7    ,$LTH               ,$PW             ,$NOE          ,GI
      COMMON                    NOT1SV          ,NOS1SV        ,LMTS2
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)
      DATA KOUNT5 /0/, UBLANK/'    '/
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS
 1000 FORMAT(20A4/5I4,A4,I4)
      KEYXC=0
      IF(UC.NE.UBLANK) KEYXC=1
      CALL PAGE
      CALL LINE(10)
      KOUNT5=KOUNT5+1
      PRINT 2, KOUNT5
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 )
      CALL LINE(4)
      PRINT 1005, UHEAD
 1005 FORMAT(25X,20A4)
      CALL LINE(4)
      PRINT 10, N,M,A,B
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //
     1        30X,'NUMBER OF FUNCTIONS =',I4 //
     2        30X,'COST COEFFICIENT A  =',I4//
     3        47X,                 'B  =',I4)
      CALL LINE(1)
      IF(KEYXC.NE.0) GO TO 25
      PRINT 21
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')
      GO TO 30
   25 CONTINUE
      PRINT 28
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES
     1 X, Y ---')
   30 CONTINUE
      CALL LINE(5)
C***** SET UP EXTERNAL VARIABLES *****
      N2=2**N
      IF(NEPMAX.EQ.0)NEPMAX = N2/2
      H=N*N2
      J=N2
      L= 1
      I=0
      DO 1011 II=1,N
       J=J/2
       L=L*2
       SN= 1
       DO 1010 LL=1,L
```

```
        SN=-SN
        V=(1+SN)/2
        DO 1009 JJ=1,J
         I=I+1
         P$(1,I)=V
      IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009   CONTINUE
 1010  CCNTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,   CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DC 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWORK     COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      S = 0
      T = 0
      CALL UNNECE
      GATES = M
      C = 0
      DO 4 GI =  1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM)GOTO4
      IF(LISUCC(GI).GT.0)GATES=GATES+1
    4 CONTINUE
      OLDCST = A*GATES + B*(C)
      T=0
      S=0
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
```

```
C**** PROCEDURE NTCD  (PROCII)       **************************************
      CALL PROCII(3,1,0,0)
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORK DERIVED BY PROCII')
      PRINT 3916,TIME
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      GATES = M
      C = 0
      DO 36 GI = 1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM) GO TO 36
      IF(LISUCC(GI).GT.0) GATES = GATES + 1
   36 CONTINUE
      NEWCST = A*GATES + B*C
      IF(NEWCST.LT.OLDCST)GO TO 37
      PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
   37 CALL LINE(3)
      PRINT 320,NEWCST
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCII'/9X,' COST=',I5,'.')
      GO TO 990
  500 STOP
      END


      SUBROUTINE CONCCO(GCO,CCO)
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C
C     THIS SUBROUTINE CONNECTS INPUT, CCO, CHOSEN IN BLOCK 25, TO
C        A GATE GCO.  SOME ARRAYS ARE ALSO UPDATED.
C        (CONCCO = CONNECT CCO)
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM
C
C     CTOWN,I,J,X,Y ARE JUST TEMPORARY VARIABLES USED IN THIS SUBROUTINE
C
      COMMON NEPMAX
      COMMON    N               , M              , A               , B
     1    ,     R               , N2             , N1              , NR
     2    ,     NM              , KFLAG          , JFLAG           , COST
     3    ,     LEVM            , NRN2           , NM1             , NN2
      COMMON    ISUCC(40,40)    , LISUCC(40)     , IPRED(40,40)    , LIPRED(40)
     1    ,     INC$MX(40,40)   , SUC$MX(40,40)  , P$(2,1280)      , UNAME(40)
     2    ,     GLEVEL(40)      , LGLIST(40)     , HLIST(40,40)    , TIME
      COMMON    T               , RTCONN(100)    , S               , RSCONN(100)
      COMMON    IFLAG           ,POINTA          ,ESS1S(40)        ,F$1(32)
     1    ,F$UB1                ,INPTCV(32)      ,LISTC(40)        ,POINTC
     2    ,LISTL(40)            ,POINTL          ,ORIGIN(40)       ,IPATH(40)
     3    ,POINTR               ,VF$1(32)        ,VF$UB1           ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPCTAB(40)         ,LPOTAB(40)       ,NRPLC(2)
     1    ,RPLC(2,40)           ,IDXO(32)        ,IDXOE(32)        ,IDX1(32)
```

```
     2      ,IDX1E(32)          ,SUMP(32)       ,SETT1(32)      ,NOT1           G1 03380
     3      ,SETS1(40)          ,NOS1           ,SETS(40)       ,NOS            G1 03390
     4      ,STS                ,SUMS2(32)      ,SETS2(200)     ,NOS2           G1 03400
     5      ,LIP                ,NOOE           ,KEYA           ,KEYB           G1 03410
     6      ,NOO                ,NO1            ,NO1E           ,$GT            G1 03420
     7      ,$LTH               ,$PW            ,$NOE           ,GI             G1 03430
      COMMON                    NOT1SV         ,NOS1SV         ,LMTS2           G1 03440
      X = (CCO-1)*N2                                                            G1 03450
      DO 11 J=1,F$UB1                                                           G1 03460
      I = F$1(J)                                                                G1 03470
      IF(P$(2,X+I).NE.1)GO TO 11                                                G1 03480
      INPTCV(I) = INPTCV(I) + 1                                                 G1 03490
   11 CONTINUE                                                                  G1 03500
      ENTRY CNCCO(GCO,CCO)                                                      G1 03510
C     USE OF THIS ENTRY POINT BY-PASSES UPDATE OF INPTCV                        G1 03520
C     UPDATE VF$1                                                               G1 03530
      X = (CCO-1) * N2                                                          G1 03540
      CTDWN = VF$UB1 + 1                                                        G1 03550
    1 CTDWN = CTDWN - 1                                                         G1 03560
      IF(CTDWN.EQ.0) GO TO 2                                                    G1 03570
      IF(P$(2,VF$1(CTDWN)+X).EQ.0) GO TO 1                                      G1 03580
      DO 3 I = CTDWN,VF$UB1                                                     G1 03590
    3 VF$1(I) = VF$1(I+1)                                                       G1 03600
      VF$UB1 = VF$UB1 - 1                                                       G1 03610
      GOTO1                                                                     G1 03620
    2 CONTINUE                                                                  G1 03630
C     UPDATE INC$MX                                                             G1 03640
      INC$MX(CCO,GCO) = 1                                                       G1 03650
C     UPDATE LISUCC AND ISUCC                                                   G1 03660
      X = LISUCC(CCO)                                                           G1 03670
      DO 4 I=1,X                                                                G1 03680
      IF(ISUCC(I,CCO).GT.GCO)GO TO 5                                            G1 03690
    4 CONTINUE                                                                  G1 03700
    5 CONTINUE                                                                  G1 03710
      LISUCC(CCO) = X + 1                                                       G1 03720
      DO 6 J=I,X                                                                G1 03730
      Y = X-J+I+1                                                               G1 03740
      ISUCC(Y,CCO)= ISUCC(Y-1,CCO)                                              G1 03750
    6 CONTINUE                                                                  G1 03760
      ISUCC(I,CCO) = GCO                                                        G1 03770
C     UPDATE LIPRED, IPRED                                                      G1 03780
      X = LIPRED(GCO)                                                           G1 03790
      DO 8 I=1,X                                                                G1 03800
      IF(IPRED(I,GCO).GT.CCO)GO TO 9                                            G1 03810
    8 CONTINUE                                                                  G1 03820
      I = X + 1                                                                 G1 03830
    9 LIPRED(GCO) = X + 1                                                       G1 03840
      DO 10 J=I,X                                                               G1 03850
      Y = X - J + I + 1                                                         G1 03860
      IPRED(Y,GCO) = IPRED(Y-1,GCO)                                             G1 03870
   10 CONTINUE                                                                  G1 03880
  100 IPRED(I,GCO) = CCO                                                        G1 03890
C     UPDATE SUC$MX                                                             G1 03900
      CALL SUCCES                                                               G1 03910
C     UPDATE INPTCV                                                             G1 03920
      RETURN                                                                    G1 03930
      END                                                                       G1 03940


      SUBROUTINE ELANDO(GATE)                                                   G1 03950
C     EDITION  AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAG1 03960
```

```
C
C     THIS SUBROUTINE IS TO ELIMINATE NON-ESSENTIAL INPUTS TO GATE
C        "GATE" AND TO ORDER THE REST IN "LISTL" ACCORDING TO DECREAS-
C        ING NUMBER OF ESSENTIAL ONES.
C        (ELANDO = ELIMINATE AND ORDER)
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM
C
C     VARIABLE DEFINITIONS:
C       MAX1: MAXIMUM NUMBER OF ESSENTIAL 1'S BELONGING TO ANY
C             PREDECESSOR OF 'GATE'.
C      NUMIN: NUMBER OF INPUTS TO GATE 'GATE'.
C       SAVE: NAME OF A PREDECESSOR OF 'GATE'.
C
C     CTDOWN,I,J,SAVE,X,Y,Z,XX,XY,YY ARE USED AS JUST TEMPORARY VARIABLE
C                                                                       S
      COMMON NEPMAX
      COMMON    N               , M              , A              , B
     1     ,    R               , N2             , N1             , NR
     2     ,    NM              , KFLAG          , JFLAG          , COST
     3     ,    LEVM            , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)    , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1     ,    INC$MX(40,40)   , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
     2     ,    GLEVEL(40)      , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T               , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG           ,POINTA          ,ESS1S(40)       ,F$1(32)
     1     ,F$UB1               ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2     ,LISTL(40)           ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3     ,POINTR              ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42)   ,PPOTAB(40)      ,LPOTAB(40)      ,NRPLC(2)
     1     ,RPLC(2,40)          ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2     ,IDX1E(32)           ,SUMP(32)        ,SETT1(32)       ,NOT1
     3     ,SETS1(40)           ,NOS1            ,SETS(40)        ,NOS
     4     ,STS                 ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5     ,LIP                 ,NO0E            ,KEYA            ,KEYB
     6     ,NO0                 ,NO1             ,NO1E            ,$GT
     7     ,$LTH                ,$PW             ,$NOE            ,GI
      COMMON                     NOT1SV         ,NOS1SV          ,LMTS2
C     ELIMINATE NON-ESSEN. INPUTS FIRST
      Y = 0
      X = 0
      NUMIN = LIPRED(GATE)
      IF(IPATH(GATE).EQ.0.OR.KFLAG.NE.2) GO TO 1
C     SPECIAL INSTRUCTIONS FOR PROCIV  (I. E., KFLAG = 2) :  PREFER
C     CONNECTIONS FROM SUBNETWORK S(GCO) FOR FIRST REMOVAL IF GATE
C     IS IN S.
    9 NUMIN = LIPRED(GATE)
      DO 10 I = 1,NUMIN
      SAVE = IPRED(I,GATE)
      IF(IPATH(SAVE).EQ.0) GO TO 10
      IF(ORIGIN(SAVE).EQ.0) GO TO 10
      IF(ESS1S(SAVE).NE.0) GO TO 10
      CALL RNONES(GATE,SAVE,1)
      Z = (SAVE-1)*N2
      DO13 I=1,F$UB1
      IF(INPTCV(F$1(I)).NE.1)GO TO13
      IF(P$(2,Z+F$1(I)).NE.1)GO TO13
      XX = LIPRED(GATE)
      DO14 J=1,XX
```

```
      XY = IPRED(J,GATE)
      YY = (XY-1)*N2
      IF(P$(2,YY+F$1(I)).NE.1)GO TO14
      ESS1S(XY) = ESS1S(XY)+1
      GO TO13
   14 CONTINUE
   13 CONTINUE
      GO TO 9
   10 CONTINUE
    1 IF(X.EQ.NUMIN)GO TO 2
      X = X + 1
      Y = Y + 1
      SAVE = IPRED(Y,GATE)
      IF(ORIGIN(SAVE).EQ.0)GOTO1
      IF(ESS1S(SAVE).NE.0) GO TO 1
C     IF HERE,     " SAVE "  IS A NON-ESSEN. INPUT TO GATE, SO REMOVE
      CALL RNONES(GATE,SAVE,       1)
      Y = Y - 1
C     FINISH UPDATE OF ESS1S (RNONES DOES NOT COMPLETELY UPDATE)
      Z = (SAVE-1)*N2
      DO 3 I=1,F$UB1
      IF(INPTCV(F$1(I)).NE.1)GO TO 3
      IF(P$(2,Z+F$1(I)).NE.1)GO TO 3
      XX = LIPRED(GATE)
      DO 4 J=1,XX
      XY = IPRED(J,GATE)
      YY = (XY-1)*N2
      IF(P$(2,YY+F$1(I)).NE.1)GO TO 4
      ESS1S(XY) = ESS1S(XY)+1
      GO TO 3
    4 CONTINUE
    3 CONTINUE
      GO TO 1
    2 CONTINUE
      ENTRY ORDERL(GATE)
      IF(KFLAG.EQ.3)RETURN
C     NOW ORDER REMAINING INPUTS IN "LISTL"
      POINTL = 0
      MAX1 = 0
C       FIRST FIND MAX NUMBER OF ESSENTIAL ONES
      X = LIPRED(GATE)
      DO 5 I=1,X
      IF(ORIGIN(IPRED(I,GATE)).EQ.0)GO TO 5
      Y= ESS1S(IPRED(I,GATE))
      IF(Y.LE.MAX1)GO TO 5
      MAX1 = Y
    5 CONTINUE
      IF(MAX1.EQ.0)GOTO8
      CTDOWN = MAX1 + 1
C       NOW SEARCH FOR INPUTS WITH SUCCESSIVELY SMALLER NUMBERS
C       OF ESSENTIAL ONES
      DO 6 I=1,MAX1
      CTDOWN = CTDOWN - 1
      DO 7 J=1,X
      IF(ORIGIN(IPRED(J,GATE)).EQ.0)GO TO 7
      Y = ESS1S(IPRED(J,GATE))
      IF(Y.NE.CTDOWN)GO TO 7
      POINTL = POINTL + 1
      LISTL(POINTL)= IPRED(J,GATE)
    7 CONTINUE
    6 CONTINUE
```

```
 8 CONTINUE
   RETURN
   END


   SUBROUTINE MINI2(IMPROV)
C  EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C  THE NAME ATTEMPTS TO INDICATE THAT THIS SUBROUTINE IS A MINIATURE
C  VERSION OF PROCEDURE II (PROCII) - ACTUALLY, THIS ROUTINE ONLY
C  REMOVES CONNECTIONS, NONE ARE ADDED
   IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C  DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C  VARIABLE DEFINITIONS:
C  BESTSL: NAME OF A PRIORITY CANDIDATE TO DISCONNECT FROM GATE GCO.
C  CHOICE: NAME OF A GATE CHOSEN TO BECOME A COVER.
C  COMPNT: A COMPONENT OF AN INTERMEDIATE CSPF VECTOR.
C   EFLAG: SIGNALS WHICH ENTRY POINT USED.
C  FEEDGT: A GATE FEEDING GATE 'GATE'.
C   F$UBO: NUMBER OF 'NECESSARY ZEROS' LISTED IN F$0.
C     F$0: LISTS (CONSECUTIVELY) POSITIONS OF NECESSARY ZEROS IN A
C          CONNECTABLE FUNCTION VECTOR.
C    GATE: NAME OF A GATE.
C  GCOUNT: A COUNTER.
C  GORDER: A SPECIAL ORDERING OF GATES AND EXTERNAL VARIABLES SUCH
C          THAT NO GATE SUCCEEDS A PREDECESSOR IN THE ORDERING.
C  MARKED: MARKED(GI)=1 FOR GI FEEDING 'GATE' INDICATES THAT GI HAS
C          ALREADY BEEN ASSIGNED NECESSARY ZEROS CORRESPONDING TO
C          '1' COMPONENTS IN THE CSPF VECTOR FOR 'GATE'.
C  NMINLV: NUMBER OF GATES IN A CERTAIN LEVEL OF THE NETWORK.
C  SELECT: NAME OF AN INPUT SELECTED AS A CANDIDATE FOR DISCONNECTION
C          FROM GATE 'GCO'.
C       T: COUNTS REMOVED CONNECTIONS.
C  TORDER: A SPECIAL ORDERING OF GATES AND EXTERNAL VARIABLES SUCH
C          THAT VARIABLES COME FIRST FOLLOWED BY GATES WITH DECREASED
C          NUMBERS OF OUTPUTS (TIES ARE BROKEN BY GORDER).
C  TPOINT: POINTER TO TORDER.
C  T1PRED: LIST OF GCO'S PREDECESSORS AT ONE STAGE OF COMPUTATION.
C  T2PRED: LIST OF GCO'S PREDECESSORS AT ONE STAGE OF COMPUTATION.
C   T1SUB: A POINTER TO T1PRED.
C   T2SUB: A POINTER TO T2PRED.
C    USED: USED(GI)=1 MEANS GI IS AN OUTPUT GATE, OR IS A COVER FOR
C          SOME 0-COMPONENT OF 'GATE'. (IT ALSO HAS A TEMPORARY USE
C          IN BEGINNING OF PROGRAM.)
C
C  COUNT,I,II,J,K,L,MOST,Q,TCOUNT,X,XX,Y  ARE USED AS JUST TEMPORARY
C                                                           VARIABLES.
C  HOW TO INCREASE CAPACITY OF SUBROUTINE.
C  DIMENSION: T1PRED(X),T2PRED(X),GORDER(X),
C             MARKED(X),USED(X),TORDER(X)     - X EQUAL TO MAX NUMBER
C                                               OF GATES PLUS EXTERNAL
C                                               VARIABLES.
C             F$0(Y) - Y EQUAL TO: 2**(MAX ALLOWED NO. OF EX. VAR.)
C
   COMMON NEPMAX
   COMMON    N           , M            , A            , B
 1      ,    R           , N2           , N1           , NR
 2      ,    NM          , KFLAG        , JFLAG        , COST
 3      ,    LEVM        , NRN2         , NM1          , NN2
   COMMON    ISUCC(40,40) , LISUCC(40)   , IPRED(40,40) , LIPRED(40)
```

```
     1       ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2       ,    GLEVEL(40)    , LGLIST(40)    , HLIST(40,40) , TIME
      COMMON    T                , RTCONN(100)   , S            , RSCONN(100)
      COMMON    IFLAG            ,POINTA         ,ESS1S(40)     ,F$1(32)
     1      ,F$UB1              ,INPTCV(32)     ,LISTC(40)     ,POINTC
     2      ,LISTL(40)          ,POINTL         ,ORIGIN(40)    ,IPATH(40)
     3      ,POINTR             ,VF$1(32)       ,VF$UB1        ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)       ,LPOTAB(40)    ,NRPLC(2)
     1      ,RPLC(2,40)         ,IDX0(32)       ,IDX0E(32)     ,IDX1(32)
     2      ,IDX1E(32)          ,SUMP(32)       ,SETT1(32)     ,NOT1
     3      ,SETS1(40)          ,NOS1           ,SETS(40)      ,NOS
     4      ,STS                ,SUMS2(32)      ,SETS2(200)    ,NOS2
     5      ,LIP                ,NOOE           ,KEYA          ,KEYB
     6      ,NOO                ,NO1            ,NO1E          ,$GT
     7      ,$LTH               ,$PW            ,$NOE          ,GI
      COMMON                     NOT1SV         ,NOS1SV        ,LMTS2
      DIMENSION T1PRED(40),T2PRED(40),GORDER(40),F$0(32),MARKED(40)
      DIMENSION USED(40),TORDER(40)
      IMPROV = 0
      T = 0
C     ORDER GATES IN GORDER
      EFLAG = 0
      GO TO 63
C     THIS ENTRY POINT FOR CALCULATION OF GORDER ONLY
      ENTRY FORMGO
      EFLAG = 1
   63 CONTINUE
      COUNT = 0
      DO 1 I=1,LEVM
      NMINLV = LGLIST(I)
      IF(NMINLV.EQ.0)GOTO1
      DO 2 J=1,NMINLV
      COUNT = COUNT + 1
      GORDER(COUNT) = HLIST(J,I)
    2 CONTINUE
    1 CONTINUE
      IF(EFLAG.EQ.1)RETURN
C     CALCULATE NUMBER OF OUTPUTS OF EACH GATE
C     (THE ARRAY 'USED' IS USED HERE JUST TEMPORARILY)
      DO 51 I=N1,NR
      TCOUNT = 0
      DO 52 J=1,NR
      IF(INC$MX(I,J).EQ.1)TCOUNT = TCOUNT + 1
   52 CONTINUE
C     TCOUNT NOW CONTAINS  THE NUMBER OF OUTPUTS OF GATE I
      USED(I) = TCOUNT
   51 CONTINUE
      MOST = 0
      DO 53 I =N1,NR
      IF(USED(I).GT.MOST)MOST = USED(I)
   53 CONTINUE
      DO 56 I= 1,N
   56 TORDER(I) = I
      TPOINT = N1
      MOST = MOST + 1
   50 MOST = MOST - 1
      IF(MOST.LT.0)GO TO 54
      DO 55 I=1,NR
      II = GORDER(I)
      IF(II.LE.N)GO TO 55
      IF(USED(II).NE.MOST)GO TO 55
```

```
      TORDER(TPOINT) = II
      TPOINT = TPOINT + 1
   55 CONTINUE
      GO TO 50
   54 CONTINUE
C     INITIALIZE GSMALL
      DO 4 I=N1,NM
      X = (I-1)*N2
      DO 4 J=1,N2
      Y = P$(1,X+J)
      IF(Y.EQ.0)GSMALL(I,J) = -100
      IF(Y.EQ.1)GSMALL(I,J) = 1
      IF(Y.EQ.-1)GSMALL(I,J)=0
    4 CONTINUE
      EFLAG = 0
      GO TO 57
      ENTRY INITGS
      EFLAG = 1
   57 DO 3 I=1,NR
      USED(I) = 0
      IF(I.LT.N1)GO TO 58
      IF(I.GT.NM) GO TO 58
      GO TO 3
   58 DO 59 J = 1,N2
   59 GSMALL(I,J)= 0
    3 CONTINUE
      DO 62 I = N1,NM
      USED(I) = 1
   62 CONTINUE
C     INITIALIZATION
      DO 34 I=1,NR
      GATE = GORDER(I)
      IF(GATE.LT.N1)GO TO 34
      XX= LIPRED(GATE)
      IF(XX.EQ.0)GOTO34
      F$UB1 = 0
      F$UB0 = 0
      DO 35 J=1,N2
      COMPNT = GSMALL(GATE,J)
      IF(COMPNT.EQ.0)GO TO 35
      IF(COMPNT.LT.0)GO TO 36
      IF(COMPNT.GE.1000) GO TO 35
      F$UB0 = F$UB0 + 1
      F$0(F$UB0) = J
      GO TO 35
   36 IF(COMPNT.LE.-1000) GO TO 35
      F$UB1 = F$UB1 + 1
      F$1(F$UB1) = J
   35 CONTINUE
      IF(F$UB1.EQ.0)GO TO 34
      DO 38 K=1,XX
      FEEDGT = IPRED(K,GATE)
      X = (FEEDGT-1)*N2
      DO 39 L=1,F$UB1
      Y = F$1(L)
      IF(P$(1,X+Y).LE.0)GO TO 39
      IF(GSMALL(FEEDGT,Y).GT.1000)GOTO39
      IF(GSMALL(GATE,Y).EQ.-200)GOTO39
      IF(GSMALL(GATE,Y).EQ.-100)GO TO 40
      GSMALL(GATE,Y) = -200
      GO TO 39
```

```
   40 GSMALL(GATE,Y) = -FEEDGT
   39 CONTINUE
   38 CONTINUE
      DO 60 K=1,XX
   60 MARKED(IPRED(K,GATE)) = 0
      DO 41 K=1,F$UB1
      X = GSMALL(GATE,F$1(K))
      IF(X.EQ.-100)GO TO 41
      IF(X.EQ.-200)GOTO41
      X = -X
      GSMALL(+X,F$1(K))=1
      USED(X) = 1
      IF(MARKED(X).EQ.1)GOTO41
      MARKED(X) = 1
      DO 42 L=1,F$UB0
      Y = GSMALL(X,F$0(L))
      IF(Y.GT.1000.OR.Y.LT.-1000)GO TO 42
      GSMALL(+X,F$0(L))=-100
   42 CONTINUE
   41 CONTINUE
   34 CONTINUE
      IF(EFLAG.EQ.1)RETURN
C     INITIALIZE COUNTER TO LOOP ONCE FOR EACH GATE
      GCOUNT = 0
C     INCREMENT GCOUNT
    5 GCOUNT = GCOUNT + 1
C     ARE ALL GATES EXHAUSTED?
      IF(GCOUNT.LE.NR)GO TO 6
      IF(T.GT.0) IMPROV = 1
      IF(IMPROV.EQ.0)RETURN
C     IF HERE, NETWORK WAS ALTERED, SO UPDATE ARRAYS
      CALL SUBNET
      CALL PVALUE
      RETURN
    6 GCO = GORDER(GCOUNT)
C     IS GCO AN ISOLATED GATE OR EXTERNAL VARIABLE?
      IF(GCO.LE.N)GOTO5
      DO 8 I=1,N2
      IF(GSMALL(GCO,I).GE.1)GOTO7
    8 CONTINUE
C     IF HERE, GATE IS ISOLATED - REMOVE INPUTS
      X = LIPRED(GCO)
      IF(X.EQ.0)GOTO5
      DO 9 I=1,X
      Y = IPRED(I,GCO)
      INC$MX(Y,GCO) = 0
C     RECORD THE DISCONNECTION
      T = T + 1
    9 CONTINUE
      GOTO 5
C     REMOVE UNNECESSARY CONNECTIONS TO GCO IN THE NEXT FEW SECTIONS
C
C     CALCULATE F(GCO)
    7 F$UB1 = 0
      DO 10 I=1,N2
      IF(GSMALL(GCO,I).GE.0)GOTO10
      F$UB1 = F$UB1 + 1
      F$1(F$UB1) = I
   10 CONTINUE
      DO 11 I=1,F$UB1
   11 INPTCV(F$1(I)) = 0
```

```
      X = LIPRED(GCO)
      DO 222I=1,X
      ESS1S(IPRED(I,GCO)) = 0
  222 CONTINUE
      T1SUB = 0
      T2SUB = 0
      DO 48 I = 1,NR
      IF(INC$MX(I,GCO).EQ.0)GOTO48
      T1SUB = T1SUB + 1
      T1PRED(T1SUB) = I
   48 CONTINUE
   17 DO 18 I=1,X
      Y = (T1PRED(I)-1)*N2
      DO 19 J=1,F$UB1
      Q = F$1(J)
      IF(P$(1,Y+Q).NE.1)GO TO 19
      IF(INPTCV(Q).LE.0) GO TO 20
      INPTCV(Q) = INPTCV(Q) + 1
      GO TO 19
   20 IF(INPTCV(Q).LT.0)GO TO 21
      INPTCV(Q) = -T1PRED(I)
      GO TO 19
   21 INPTCV(Q) = 2
   19 CONTINUE
   18 CONTINUE
C     MARK ESSENTIAL 1'S
      DO 22 I=1,F$UB1
      Q = INPTCV(F$1(I))
      IF(Q.GE.0)GO TO 22
      ESS1S(-Q) = ESS1S(-Q) + 1
   22 CONTINUE
   46 SELECT = 0
      BESTSL = 0
      DO 45 L=1,X
      Q = T1PRED(L)
      IF(INC$MX(Q,GCO).EQ.0)GOTO45
      IF(ESS1S(Q).GT.0)GOTO45
      IF(SELECT.EQ.0)SELECT = Q
      IF(USED(Q).EQ.1)GOTO45
      IF(BESTSL.NE.0)GOTO45
      BESTSL = Q
   45 CONTINUE
      IF(SELECT.EQ.0)GO TO 47
      Q = SELECT
      IF(BESTSL.NE.0)Q = BESTSL
C     IF HERE, GATE HAS NO ESSENTIAL 1'S  -  REMOVE IT
      INC$MX(Q,GCO) = 0
      T = T + 1
C     UPDATE ESS1S
      Y = (Q - 1)*N2
      DO 24 J=1,F$UB1
      V = F$1(J)
      IF(P$(1,Y+V).NE.1)GO TO 24
C     UPDATE INPTCV FOR COMPONENT V
      INPTCV(V) = INPTCV(V) - 1
      IF(INPTCV(V).GT.1)GO TO 24
C     CASE WHEN NEW ESSEN 1 CREATED
      DO 27 K = 1,X
      W = T1PRED(K)
      IF(INC$MX(W,GCO).EQ.0) GO TO 27
      Z = (W - 1) * N2
```

```
      IF(P$(1,Z+V).EQ.0)GO TO 27
      ESS1S(W) = ESS1S(W) + 1
C     IN THIS CASE, NO NEED TO UPDATE INPTCV FURTHER
      GSMALL(GCO,V) = -W
      GO TO 24
   27 CONTINUE
   24 CONTINUE
      GOTO46
   47 DO 49 I = 1,NR
      IF(INC$MX(I,GCO).EQ.0)GOTO49
      T2SUB = T2SUB + 1
      T2PRED(T2SUB) = I
   49 CONTINUE
C     NOW ALL CURRENT INPUTS HAVE ESSENTIAL 1'S
C     INPUTS STILL CONNECTED TO GCO ARE LISTED IN T2PRED IN REVERSE
C     ORDER
C
C     UPDATE G(I)'S OF THOSE GATES STILL CONNECTED TO GATE GCO
C
      DO 29 II=1,F$UB1
      I = F$1(II)
      CHOICE = -GSMALL(GCO,I)
      IF(CHOICE.LT.100)GO TO 61
      CHOICE = 0
      DO 30 JJJ=1,NR
      JJ = TORDER(JJJ)
      IF(INC$MX(JJ,GCO).EQ.0)GO TO 30
      IF(P$(1,(JJ-1)*N2+I).NE.1)GO TO 30
      IF(JJ.LE.N)GO TO 29
      IF(CHOICE.EQ.0)CHOICE=JJ
      IF(GSMALL(JJ,I).GE.1)GOTO29
   30 CONTINUE
   61 GSMALL(CHOICE,I) = 1
      USED(CHOICE) = 1
   29 CONTINUE
      DO 32 I=1,N2
      IF(GSMALL(GCO,I).LT.1)GO TO 32
      DO 33 J=1,T2SUB
      IF(GSMALL(T2PRED(J),I).EQ.0)GSMALL(T2PRED(J),I)=-100
   33 CONTINUE
   32 CONTINUE
      GOTO5
      END


      SUBROUTINE OUTPUT(MATRIX,ARRAY)
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N              , M              , A              , B
     1     ,    R              , N2             , N1             , NR
     2     ,    NM             , KFLAG          , JFLAG          , COST
     3     ,    LEVM           , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1     ,    INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
     2     ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
     1     ,F$UB1              ,INPTCV(32)      ,LISTC(40)       ,POINTC
```

```
     2      ,LISTL(40)        ,POINTL         ,ORIGIN(40)     ,IPATH(40)
     3      ,POINTR           ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)
      COMMON   POTAB(200,42),PPOTAB(40)       ,LPOTAB(40)     ,NRPLC(2)
     1      ,RPLC(2,40)       ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)
     2      ,IDX1E(32)        ,SUMP(32)       ,SETT1(32)      ,NOT1
     3      ,SETS1(40)        ,NOS1           ,SETS(40)       ,NOS
     4      ,STS              ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5      ,LIP              ,NOOE           ,KEYA           ,KEYB
     6      ,NOO              ,NO1            ,NO1E           ,$GT
     7      ,$LTH             ,$PW            ,$NOE           ,GI
      COMMON                  NOT1SV         ,NOS1SV         ,LMTS2
      DIMENSION UX(5), UY(5), UG(40), UF(40), ARRAY(40), ARRAY2(2,1280)
      DIMENSION MATRIX(40,40)
      DATA UX /' X1',' X2',' X3',' X4',' X5'/
      DATA UY/' Y1',' Y2',' Y3',' Y4',' Y5'/
      DATA UF /'  1','  2','  3','  4','  5','  6','  7','  8'
     1       ,  '  9',' 10',' 11',' 12',' 13',' 14',' 15',' 16'
     2       ,  ' 17',' 18',' 19',' 20',' 21',' 22',' 23',' 24'
     3       ,  ' 25',' 26',' 27',' 28',' 29',' 30',' 31',' 32'
     4       ,  ' 33',' 34',' 35',' 36',' 37',' 38',' 39',' 40'/
      DATA GMAX/40/
C
      KEYXC=ARRAY(1)
      IF(KEYXC.NE.0) GO TO 50
      DO 1 GI=1,N
       UNAME(GI)=UX(GI)
    1 CONTINUE
      GO TO 100
   50 CONTINUE
      L=N/2
      DO 4 GI=1,L
       UNAME(GI)=UX(GI)
      UNAME(GI+L)=UY(GI)
    4 CONTINUE
  100 CONTINUE
      DO 2 GI=N1,GMAX
       UNAME(GI)=UF(GI-N)
    2 CONTINUE
      RETURN
C
      ENTRY LINE(L)
      DO 6 LL=1,L
       PRINT 5
    5 FORMAT(1H )
    6 CONTINUE
      RETURN
C
      ENTRY PAGE
      PRINT 7
    7 FORMAT(1H1)
      RETURN
C
      ENTRY CKT(MATRIX,ARRAY)
      PRINT 10
   10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY'/)
      DO 20 GJ=N1,NR
       G=0
       DO 15 GI=1,NR
        IF(MATRIX(GI,GJ).EQ.0) GO TO 15
        G=G+1
        UG(G)=UNAME(GI)
```

```
 15  CONTINUE
     IF(G.EQ.0) GO TO 18
     PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)
 17  FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35(    A3))
     GO TO 20
 18  PRINT 19, UNAME(GJ),ARRAY(GJ)
 19  FORMAT(1H0, 9X,A3,5X,'/',I2,'/')
 20 CONTINUE
    RETURN
C
    ENTRY TRUTH(ARRAY2,J)
    IF(J.EQ.2) GO TO 36
    PRINT 35
 35 FORMAT(11X,   'TRUTH TABLE'/)
    GO TO 38
 36 PRINT 37
 37 FORMAT(11X,  'REQUIREMENT TABLE')
 38 CONTINUE
    DO 40 GI=1,NR
     ILO=(GI-1)*N2+1
     IHI=ILO+N2-1
     PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)
 40 CONTINUE
 41 FORMAT(1H0, 9X,A3,' = ', 32(I1,1X))
    RETURN
    END


    SUBROUTINE PROCII(JAYFLG,KEIFLG,EYEFLG,LFLAG)
C   EDITION  BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C
C   JAYFLG INDICATES THE VERSION OF ORDERING USED
C   KEIFLG=1 INDICATES NORMAL PROCII
C   KEIFLG = 2 INDICATES PROCIV (VERSION B)
C   KEIFLG = 3 INDICATES PROCIP
C   EYEFLG INDICATES THE GATE OF FOCUS FOR PROCIV
C   SET EYEFLG = 0 FOR OTHER PROCEDURES
C   LFLAG IS USED FOR PROCIV AND PROC3
C   LFLAG = 1 MEANS IGNORE GATES OF SUBNETWORK T
C   LFLAG = 0 MEANS CALCULATE CSPF'S FOR GATES OF SUBNETWORK T
C   SET LFLAG = 0 FOR PROCII AND PROCIP
C   LFLAG = -1 MEANS USE SPECIAL ORDERINGS PREFERING HIGH LEVEL GATES
C   GESTED FOR PROC3 - DEFINITELY NOT FOR PROCIV)
C
    IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C   DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C   VARIABLE DEFINITIONS:
C        C: COMPONENT POSITION OF A '0' IN GCO'S CSPF.
C   CAFTER: NO. OF CONNECTIONS AFTER TRANSFORMATICN.
C   CBEFOR: NO. OF CONNECTIONS BEFORE TRANSFORMATION.
C      CCO: AN EFFECTIVELY CONNECTIBLE FUNCTION CHOSEN TO BE ADDED TO
C           GCO AS A NEW INPUT.
C   CHOICE: AN INPUT CHOSEN TO COVER A '0' COMPONENT.
C   COVERD: COVERD(GI) TELLS NO. OF 0'S IN CSPF VECTOR OF GCO COVERED
C           BY GI.
C      CVD: NO. OF ZEROS IN CSPF VECTOR OF GCO COVERED BY AN INPUT I.
C   EFFCON: A SELECTED EFFECTIVELY CONNECTIBLE FUNCTION.
C   ESTEMP: TEMPORARY STORAGE USED TO CALCULATE 'ESSIS'.
C    F$UBO: NUMBER OF NECESSARY ZEROS LISTED IN F$O.
```

```
C        F$0: LISTS (CONSECUTIVELY) POSITIONS OF NECESSARY ZEROS IN A
C             CONNECTIBLE FN.
C     GAFTER: NO. OF GATES IN NETWORK AFTER TRANSFORMATION.
C     GBEFOR: NO. OF GATES IN NETWORK BEFORE TRANSFORMATION.
C        GCO: THE GATE (OR EX. VAR.) IN GORDER POINTED TO BY GCOUNT.
C     GCOUNT: A POINTER TO CURRENT POSITION IN GORDER.
C     GORDER: CONTAINS AN ORDERED LIST OF GATES AND EXTERNAL VARIABLES.
C     IMPROV: PARAMETER RETURNED BY MINI2 (UNUSED BY PROCII).
C     INSERT: POSITION IN LISTC TO INSERT NEWCOMERS.
C       IPOS: POINTS TO GATE EYEFLG IN GORDER (FOR PROCIV).
C     IPOSP1: EQUALS IPOS+1.
C     NEWCST: COST OF NEW NETWORK.
C     NMINLV: NO. OF GATES AND EX. VAR. IN A CERTAIN LEVEL OF NETWORK.
C     OLDCST: COST OF ORIGINAL NETWORK.
C     ORDERP: ORDERING OF GCO'S INPUTS ONLY - IN SAME ORDER AS RORDER.
C          P: A CONNECTIBLE FN. TO REPLACE INPUT TH TO GCO.
C      PICK1: FIRST PREFERENCE COVER.
C      PICK2: SECOND PREFERENCE COVER.
C      PICK3: THIRD PREFERENCE COVER.
C      PICK4: FOURTH PREFERENCE COVER.
C      POINT: POINTS TO LAST ENTRY IN ORDERP.
C     POINTT: NO. OF ELEMENTS IN THESS.
C     QINC$M: SAVES A COPY OF ORIGINAL INC$MX.
C     RORDER: AN ORDERING OF GATES AND EX. VARS. DERIVED FROM GORDER
C             (STRUCTURE OF ORDERING DIFFERS ACCORDING TO KEIFLG AND
C             LFLAG).
C     SAVING: COST IMPROVEMENT OF NEW NETWORK OVER ORIGINAL (CAN BE
C             POSITIVE OR NEGATIVE).
C       SUBL: POINTER TO ELEMENTS OF LISTL.
C         TH: AN INPUT TO GCO SELECTED FOR POSSIBLE REPLACEMENT.
C      THESS: LIST OF ESSENTIAL 1'S IN A PARTICULAR INPUT (TH) TO GCO.
C     THROWN: EQUAL TO NO. OF ENTRIES THROWN OUT OF LISTC WHEN A
C             CERTAIN NEW ENTRY IS CONSIDERED FOR INSERTION.
C
C     COUNT,I,II,J,JJ,K,KK,LAST,TEMP,TEND,TEST,X,XY,Y,Z,ZFLAG,ZONE,ZZ
C     ARE USED AS JUST TEMPORARY VARIABLES.
C
C     HOW TO INCREASE CAPACITY OF SUBROUTINE.
C     DIMENSION: COVERD(X),RORDER(X),
C                QINC$M(X,X),ORDERP(X) - X EQUAL TO MAX NUMBER OF GATES
C                                        PLUS EXTERNAL VARIABLES.
C                GORDER(Y) - Y EQUAL TO X+1
C                ESTEMP(Z),F$0(Z),THESS(Z) - Z EQUAL TO 2**(MAX ALLOWED
C                                            NO. OF EX. VAR.)
C
C     **** NOTE PROGRAM SECTIONS DO NOT CORRESPOND EXACTLY IN NAME OR
C          NUMBER TO FLOWCHART BLOCKS IN USER'S MANUAL. *****
C
C
      COMMON NEPMAX
      COMMON   N             , M             , A             , B
     1   ,     R             , N2            , N1            , NR
     2   ,     NM            , KFLAG         , JFLAG         , COST
     3   ,     LEVM          , NRN2          , NM1           , NN2
      COMMON   ISUCC(40,40)  , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)
     1   ,     INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2   ,     GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)  , TIME
      COMMON   T             , RTCONN(100)   , S             , RSCONN(100)
      COMMON   IFLAG         ,POINTA         ,ESS1S(40)      ,F$1(32)
     1   ,F$UB1              ,INPTCV(32)     ,LISTC(40)      ,POINTC
     2   ,LISTL(40)          ,POINTL         ,ORIGIN(40)     ,IPATH(40)
```

```
    3     ,POINTR          ,VF$1(32)        ,VF$UB1         ,GSMALL(40,32)
     COMMON   POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2)
    1     ,RPLC(2,40)      ,IDXO(32)        ,IDXOE(32)      ,IDX1(32)
    2     ,IDX1E(32)       ,SUMP(32)        ,SETT1(32)      ,NOT1
    3     ,SETS1(40)       ,NOS1            ,SETS(40)       ,NOS
    4     ,STS             ,SUMS2(32)       ,SETS2(200)     ,NOS2
    5     ,LIP             ,NOOE            ,KEYA           ,KEYB
    6     ,NOO             ,NO1             ,NO1E           ,$GT
    7     ,$LTH            ,$PW             ,$NOE           ,GI
     COMMON                NOT1SV           ,NOS1SV         ,LMTS2
     DIMENSION              COVERD(40)      ,ESTEMP(32)     ,F$0(32)
    1     ,GORDER(41)      ,RORDER(40)      ,QINC$M(40,40)  ,THESS(32)
    2     ,ORDERP(40)
     IF(KEIFLG.NE.2)GOTO 185
     IF(LISUCC(EYEFLG).EQ.0)RETURN
C
C    SECTION 1, START
C
 185 CALL SUBNET
     CALL PVALUE
     CALL UNNECE
     KFLAG = KEIFLG
     JFLAG = JAYFLG
     IFLAG = EYEFLG
C    CALCULATE NUMBER OF GATES ACTUALLY IN NETWORK
     GBEFOR = M
     CBEFOR = 0
     DO 167 I=1,NR
     CBEFOR = CBEFOR + LISUCC(I)
     IF(I.LE.NM) GO TO 167
     IF(LISUCC(I).GT.0) GBEFOR = GBEFOR + 1
 167 CONTINUE
     IF(KFLAG.NE.2)GOTO186
     DO 187 I=1,NR
     IPATH(I)=0
     IF(SUC$MX(IFLAG,I).GT.0)IPATH(I)=1
 187 CONTINUE
     IPATH(IFLAG)=1
C    MAKE A COPY OF INC$MX IN "QINC$M"
 186 DO 132 I=1,NR
C    INITIALIZE GSMALL WITH DON'T CARES -
C        GSMALL RECORDS FOR EACH GATE THE SUMMATION OF COMPONENTS
C        OF ALL FUNCTIONS (G(I,J)) REQUIRED BY LINES FROM GATE I
C        TO EVERY SUCCESSOR J.
     DO 50 J=1,N2
  50 GSMALL (I,J)= 0
     DO 132 J=1,NR
 132 QINC$M(I,J)=INC$MX(I,J)
C
C    SECTION 2, ORDER GATES AND EXTERNAL VARIABLES
C
   2 ZONE = 0
     COUNT=1
 189 DO 101 I=1,LEVM
     NMINLV=LGLIST(I)
     IF(NMINLV.EQ.0)GOTO101
     DO 102 J=1,NMINLV
     K = HLIST(J,I)
     IF(KFLAG.EQ.2.AND.IPATH(K).EQ.ZONE) GO TO 102
     GORDER(COUNT)=K
     IF(K .EQ.IFLAG) GO TO 133
```

```
      COUNT=COUNT+1
  102 CONTINUE
  101 CONTINUE
      IF(COUNT.EQ.NR+1)GOTO160
  133 GORDER(COUNT)=IFLAG
      IPOS = COUNT
       COUNT = COUNT + 1
      ZONE = 1
      GOTO 189
C     FORM RORDER
  160 IF(KFLAG.EQ.2) GO TO 161
      IF(LFLAG.EQ.-1) GO TO 214
      DO 162 I = 1,NR
  162 RORDER(I) = GORDER(I)
      GO TO 3
  214 DO 215 I = 1,NR
  215 RORDER(I) = GORDER(NR+1-I)
      GOTO 3
  161 IPOSP1 = IPOS + 1
      DO 163 I = IPOSP1,NR
  163 RORDER(I - IPOS) = GORDER(I)
      J = NR - IPOS
      DO 169 I = 1,IPOS
  169 RORDER(J+I) = GORDER(I)
C
C     SECTION 3, CALCULATE F(I)
C
    3 LAST=N2*NR
      DO 104 I=1,LAST
  104 P$(2,I)=P$(1,I)
C     INITIALIZE GSMALL FOR OUTPUT GATES
      DO 53 I=N1,NM
      X=(I-1)*N2
      DO 53 J=1,N2
      Y = P$(2,X+J)
      IF(Y.EQ.0) GSMALL(I,J) = -100
      IF(Y.EQ.1) GSMALL(I,J) = 1
      IF(Y.EQ.-1) GSMALL(I,J) = 0
   53 CONTINUE
C
C     SECTION 4, INITIALIZE COUNTER TO LOOP ONCE FOR EACH GATE
C
    4 GCOUNT=0
C
C     SECTION 5, INCREMENT COUNT
C
5     GCOUNT=GCOUNT+1
C
C     SECTION 6, ARE ALL GATES EXHAUSTED?
C
    6 IF(LFLAG.EQ.1.AND.GCOUNT.EQ.IPOS)GOTO7
      IF(GCOUNT.GT.NR)GOTO7
      GCO=GORDER(GCOUNT)
      IF(GCO.LE.N) GO TO 33
C
C     SECTION 11,CALCULATE G(GCO)
C
11    X = N2*(GCO-1)
      DO 190 I=1,N2
      Y = GSMALL(GCO,I)
      IF(Y.EQ.0) P$(2,X+I) = -1
```

```
      IF(Y.GT.0) P$(2,X+I) = 1
      IF(Y.LT.0) P$(2,X+I) = 0
  190 CONTINUE
C
C     SECTION 33, IS GCO AN ISOLATED GATE OR AN EXTERNAL VARIABLE ?
C       IF ISOLATED: REMOVE ANY INPUTS, UPDATE ARRAYS, RETURN TO SEC. 5
C       IF EXTERNAL VARIABLE: RETURN TO SECTION 5
C
   33 IF(GCO.LE.N)GO TO 5
      DO 54 I=1,N2
      IF(GSMALL(GCO,I).GE.1)GOTO12
   54 CONTINUE
C     IF HERE, THEN GATE IS NONESSENTIAL - REMOVE INPUTS AND OUTPUTS
      X = LISUCC(GCO)
      IF(X.EQ.0)GO TO 201
      DO 202 I=1,X
      Y = ISUCC(1,GCO)
      CALL RNONES(Y,GCO,0)
  202 CONTINUE
  201 X=LIPRED(GCO)
      IF(X.EQ.0)GO TO 5
      DO 55 I=1,X
      Y = IPRED(1,GCO)
      CALL RNONES(GCO,Y,0)
   55 CONTINUE
      GO TO 5
C
C     SECTION 12, CALCULATE F(GCO)
C
   12 Z=N2*(GCO-1)
      F$UB0=1
      F$UB1=1
      DO 51 I=1,N2
      X=GSMALL(GCO,I)
      IF(X.EQ. 0)GO TO 51
      IF(X.LT. 0)GO TO 52
      F$0(F$UB0)=I
      F$UB0=F$UB0+1
      GO TO 51
   52 F$1(F$UB1)=I
      VF$1( F$UB1) = I
      F$UB1=F$UB1+1
   51 CONTINUE
      F$UB0=F$UB0-1
      F$UB1=F$UB1-1
      VF$UB1 = F$UB1
C
C     SECTION 13, THIS SECTION CONTAINS SUBSECTIONS 15 THROUGH 32 + 34
C
C       THIS SECTION WILL ADD CONNECTIBLE FUNCTIONS, REMOVE ANY
C       UNNECESSARY INPUTS, UPDATE G(I,GCO) FOR ALL I, WHERE I FEEDS GCO
C
C
C     SECTION 34, CALCULATE HOW MANY INPUTS COVERING EACH COMPONENT
C
      DO 88 I=1,F$UB1
   88 INPTCV(F$1(I))=0
      X = LIPRED(GCO)
      DO 57 I=1,X
      XY = IPRED(I,GCO)
      Y = N2*(XY-1)
```

```
      DO 89 J=1,F$UB1
      Z = F$1(J)
      IF(P$(2,Y+Z).EQ.1)INPTCV(Z)=INPTCV(Z)+1
   89 CONTINUE
   57 CONTINUE
C     INITIALIZE ORIGIN(40)
      DO 150 I=1,NR
      IF(INC$MX(I,GCO).EQ.0)GO TO 151
      ORIGIN(I) = 1
      GO TO 150
  151 ORIGIN(I) = 0
  150 CONTINUE
C
C     SECTION 15, LIST NUMBER OF ESSENTIAL ONES IN EACH INPUT TO GCO
C
   15 DO 59 I=1,F$UB1
   59 ESTEMP(I)=0
      X=LIPRED(GCO)
      DO 172 J=1,X
      XY=IPRED(J,GCO)
      Y=N2*(XY-1)
      DO 56 I=1,F$UB1
      Z=F$1(I)
      TEMP=P$(2,Y+Z)
      IF(TEMP.LE.0)GOTO56
      TEST=ESTEMP(I)
      IF(TEST.LT.0)GOTO56
      IF(TEST.GT.0)GOTO58
      ESTEMP(I)=XY
      GO TO 56
   58 ESTEMP(I)=-1
   56 CONTINUE
  172 CONTINUE
      DO 60 I=1,X
   60 ESS1S(IPRED(I,GCO))=0
      DO 61 I=1,F$UB1
      Y=ESTEMP(I)
      IF(Y.LE.0)GO TO 61
      ESS1S(Y)=ESS1S(Y)+1
   61 CONTINUE
C
C     SECTION 16, ELIMINATE NON-ESSENTIAL INPUTS AND ORDER OTHERS (IN
C       "LISTL") BY DECREASING NUMBER OF ESSENTIAL ONES
C
   16 CALL ELANDO(GCO)
C
C     SECTION 17, LIST ALL CONNECTIBLE INPUTS TO GCO (IN LISTC) IN ORDER
C        OF DECREASING NUMBER OF ZEROS IN G(GCO) COVERED.  DO NOT LIST
C        ANY SMALLER THAN OTHERS.
C
C     SEARCH FOR CONNECTIBLE INPUTS THAT COVER AT LEAST ONE 0
   17 POINTC=0
      IF(KFLAG.EQ.3)GOTO32
      DO 71 I=1,NR
C     GATE I MUST NOT BE A SUCCESSOR OF GATE GCO
      IF(SUC$MX(GCO,I).GT.0)GOTO71
C     GATE I MUST NOT FEED (NOW OR PREVIOUSLY) GATE GCO
      IF(OINC$M(I,GCO).GT.0)GOTO71
      Y=N2*(I-1)
C     GATE I MUST HAVE ALL THE NECESSARY 0'S
      DO 72 J=1,F$UB0
```

```
      IF(P$(2,F$O(J)+Y).NE.0)GOTO71
   72 CONTINUE
C     GATE I MUST COVER AT LEAST ONE 0 IN GCO
      DO 73 J=1,F$UB1
      IF(P$(2,F$1(J)+Y).EQ.1)GOTO74
   73 CONTINUE
      GOTO71
   74 CONTINUE
C     GATE I MUST NOT BE ISOLATED IF NOT AN OUTPUT GATE
      IF(LISUCC(I).EQ.0.AND.(I.GT.NM           ))GOTO71
C     IF PROCIV IN EFFECT, CONNECTIBLE CANNOT BE ON
C     PATH FROM GATE IFLAG TO OUTPUT
      IF(KFLAG.EQ.2.AND.IPATH(I).EQ.1)GOTO71
C     TRY TO PLACE GATE NUMBER INTO CORRECT POSITION IN LISTC
      ZFLAG = 1
      ZZ = (I-1)*N2
      IF(POINTC.NE.0)GOTO75
C     CASE WHEN LISTC IS EMPTY
      ZFLAG = 0
      POINTC = 1
      LISTC(1)= I
C     CALCULATE NUMBER OF ZEROS IN G(GCO) COVERED BY I
   75 CVD = 0
      DO 77 J=1,F$UB1
      IF(P$(2,F$1(J)+ZZ).NE.1)GOTO77
      CVD= CVD+1
   77 CONTINUE
      COVERD(I) = CVD
      IF(ZFLAG.EQ.0)GOTO71
      INSERT = 0
      THROWN = 0
      DO 76 JJ=1,POINTC
      J = JJ - THROWN
C     TEST IF ENTRY IS .GT. OR .LT. NEW INPUT
      Y= COVERD(LISTC(J))
      Z= (LISTC(J)-1)*N2
      IF(INSERT.NE.0)GOTO78
      IF(Y.LE.CVD)INSERT = J
   78 CONTINUE
      IF(CVD.LE.Y)GO TO 79
C     TEST HERE IF VECTOR CORRES. TO J IS SMALLER THAN VEC. CORRES. TO I
      DO 81 K=1,F$UB1
      IF(P$(2,Z+F$1(K)).GT.P$(2,ZZ+F$1(K)))GO TO 76
   81 CONTINUE
      IF(INSERT.NE.J)GO TO 82
C     IF HERE, WE CAN SIMPLY EXCHANGE I WITH J
      INSERT = -1
      LISTC(J) = I
      GO TO 76
   82 TEND = POINTC-THROWN-1
      IF (TEND.EQ.0)GOTO200
      DO 83 K=J,TEND
   83 LISTC(K)= LISTC(K+1)
  200 THROWN = THROWN + 1
      GO TO 76
C     TEST HERE IF VEC. CORRES. TO J IS .EQ. TO VEC. CORRES. TO I
   79 IF(CVD.LT.Y)GO TO 80
      DO 84 K=1,F$UB1
      IF(P$(2,Z+F$1(K)).NE.P$(2,ZZ+F$1(K)))GO TO 76
   84 CONTINUE
C     IF HERE, VECTORS FOR I AND J ARE IDENTICAL, ONLY INSERT IF I IS
```

```
C        GATE AND J IS EX. VAR.
      IF(I.LE.N.OR.LISTC(J).GE.N1)GO TO 71
      LISTC(J)=I
      GOTO71
C     TEST HERE IF VECTOR CORRES. TO J LARGER THAN VEC. CORRES. TO I
   80 DO 85 K=1,F$UB1
      IF(P$(2,Z+F$1(K)).LT.P$(2,ZZ+F$1(K)))GO TO 76
   85 CONTINUE
C     IF HERE, WE CAN THROW OUT I
      GO TO 71
   76 CONTINUE
      POINTC = POINTC - THROWN
C     NOW INSERT NEW INPUT INTO LISTC
      IF(INSERT.LT.0)GO TO 71
      IF(INSERT.GT.0)GO TO 86
C     IF HERE, ADD NEW INPUT TO END OF LISTC
      POINTC = POINTC + 1
      LISTC(POINTC)=I
      GO TO 71
C     SHIFT AND INSERT
   86 POINTC = POINTC + 1
      DO 87 J=INSERT,POINTC
      JJ = POINTC-(J-INSERT)
   87 LISTC(JJ+1)=LISTC(JJ)
      LISTC(INSERT) = I
   71 CONTINUE
      IF(POINTC.EQ.0)GOTO32
C
C     SECTION 18, SELECT TOP ELEMENT, (CALL IT "TH") FROM LISTL
C
   18 IF(POINTC.EQ.0)GOTO 32
      TH = LISTL(1)
      SUBL = 1
C
C     SECTION 19, SEARCH LISTC FOR A REPLACEMENT FOR TH
C
C     LIST ESSENTIAL ONES OF TH
   19 POINTT=0
      Y = (TH-1)*N2
      DO 90 I=1,F$UB1
      X = F$1(I)
      IF(INPTCV(X).NE.1)GOTO90
      IF(P$(2,Y+X).NE.1)GOTO90
      POINTT = POINTT+1
      THESS(POINTT) = X
   90 CONTINUE
C     NOW SEARCH LISTC
      DO 91 I=1,POINTC
C     IF CURRENT ELT. OF LISTC BEING CHECKED HAS FEWER COVERING 1'S THAN
C        TH HAS ESSENTIAL 1'S, THEN NO REPLACEMENT, P, CAN BE FOUND IN
C        REMAINDER OF LISTC- SO SKIP TO SECTION 21
      IF(COVERD(LISTC(I)).LT.POINTT)GO TO 21
      Y = N2*(LISTC(I)-1)
      DO 92 J=1,POINTT
      X = THESS(J)
      IF(P$(2,Y+X).NE.1)GOTO91
   92 CONTINUE
C     IF HERE, GATE (OR EX. VAR.) LISTC(I) IS A REPLACEMENT, P, FOR TH
      P = LISTC(I)
      GO TO 23
   91 CONTINUE
```

```
      GO TO 21
C
C     SECTION 23, A REPLACEMENT (CONNECTIBLE FN.), P, HAS BEEN FOUND
C       TO REPLACE TH (CURRENTLY AN INPUT TO G(GCO) - SO DISCONNECT TH
C
   23 CALL RNONES(GCO,TH,        2)
C
C     SECTION 30, CONNECT REPLACEMENT GATE, P, TO GCO
   30 CALL CNCCO(GCO,P)
      IF(POINTL.EQ.0) GO TO 24
C     UPDATE ESS1S AND INPTCV
      DO 118 I=1,F$UB1
      C = F$1(I)
      X = (TH-1)*N2
      Y = (P -1)*N2
      IF(P$(2,C+X).EQ.1)GO TO 119
      IF(P$(2,C+Y).EQ.1)GO TO 120
C     IF HERE, NO CHANGE NEEDED FOR COMPONENT C
      GO TO 118
  120 INPTCV(C) = INPTCV(C)+1
      IF(INPTCV(C).GT.2)GO TO 118
      DO 121 J=1,POINTL
      Z = (LISTL(J)-1)*N2
      IF(P$(2,Z+C).NE.1)GO TO 121
      ESS1S(LISTL(J)) = ESS1S(LISTL(J)) - 1
      GO TO 118
  121 CONTINUE
  119 IF(P$(2,C+Y).EQ.1)GO TO 122
      IF(INPTCV(C).GT.1)GO TO 118
      DO 123 J=1,POINTL
      Z = (LISTL(J)-1)*N2
      IF(P$(2,Z+C).NE.1)GO TO 123
      ESS1S(LISTL(J)) = ESS1S(LISTL(J))+1
      GO TO 118
  123 CONTINUE
  122 INPTCV(C) = INPTCV(C) + 1
  118 CONTINUE
C
C     SECTION 24, IS LISTL EMPTY ?
C
   24 IF(POINTL.EQ.0)GO TO 32
C     IF NOT EMPTY, GO TO SECTION 29
C
C     SECTION 29, REORDER LISTL
C
   29 CALL FLANDO(GCO)
C
C     SECTION 31, REMOVE P FROM LISTC
C
   31 POINTC = POINTC - 1
      IF(POINTC.LE.0) GO TO 131
      DO 129 I=1,POINTC
      IF(LISTC(I).NE.P)GOTO129
C     COMPRESS LISTC, OVERWRITING ENTRY P
      DO 130 J=I,POINTC
      LISTC(J) = LISTC(J+1)
  130 CONTINUE
      GO TO 131
  129 CONTINUE
  131 CONTINUE
C     RETURN TO SECTION 18
```

```
      GO TO 18
C
C     SECTION 21, ARE WE AT BOTTOM OF LISTL ?
C
   21 IF( POINTL .EQ. SUBL ) GO TO 25
      GO TO 22
C
C     SECTION 22, SELECT NEXT LOWER ELEMENT IN LISTL AS NEW TH
C
   22 SUBL = SUBL + 1
      TH = LISTL(SUBL)
      GO TO 19
C
C     SECTION 25, SEARCH LISTC FOR EFFECTIVELY CONNECTIBLE FN., "EFFCON"
C
   25 IF(POINTC.EQ.0)GO TO 32
      IF(VF$UB1.EQ.0) GOTO32
      COUNT = 0
  137 IF(COUNT.EQ.POINTC)GO TO 138
      COUNT = COUNT + 1
      CCO = LISTC(COUNT)
      X=(CCO-1) * N2
      DO 139 I=1,VF$UB1
      IF(P$(2,VF$1(I)+X).EQ.1)GO TO 140
  139 CONTINUE
      GO TO 137
  140 CONTINUE
C     DOES CCO COVER ANY ESSENTIAL ONES ?
      DO 141 I=1,VF$UB1
      IF(INPTCV(VF$1(I)).NE.1)GO TO 141
      IF(P$(2,VF$1(I)+X).EQ.1)GO TO 142
  141 CONTINUE
      GOTO137
  142 CONTINUE
C     CONNECT CCO
      EFFCON = CCO
      GO TO 27
  138 CONTINUE
C     IF HERE, NO ESSENTIAL 1'S COULD BE COVERED - SO CONNECT EFFECTIVE
C        CONNECTIBLE ONE AT A TIME UNTIL NO MORE CAN BE ADDED.  THEN GO
C        TO SECTION 32 (SINCE NO MORE ORIGINAL INPUTS CAN BE ELIMINATED.
      COUNT = 0
  144 IF(COUNT.EQ.POINTC)GO TO 32
      COUNT = COUNT + 1
      CCO = LISTC(COUNT)
      X = (CCO-1) * N2
      DO 145 I=1,VF$UB1
      IF(P$(2,VF$1(I)+X).EQ.1)GO TO 146
  145 CONTINUE
      GO TO 144
  146 CONTINUE
C     CONNECT CCO, UPDATE ARRAYS (INCLUDING VF$1)
      CALL CONCCO(GCC,CCO)
      GO TO 144
C
C     SECTIONS 27 AND 28  -  CONNECT EFFCON AND UPDATE ARRAYS
C
C     UPDATE ESS1S
   27 X = (CCO-1)*N2
      DO 155 II=1,F$UB1
      I = F$1(II)
```

```
      IF(P$(2,X+I).NE.1)GO TO 155
      IF(INPTCV(I).NE.1)GO TO 155
C     IF HERE, SOME GATE WILL LOSE AN ESSENTIAL ONE
      Y = LIPRED(GCO)
      DO 156 J=1,Y
      XY = IPRED(J,GCO)
      Z = (XY-1) * N2
      IF(P$(2,Z+I).NE.1)GO TO 156
      ESS1S(XY) = ESS1S(XY) - 1
      GO TO 155
  156 CONTINUE
  155 CONTINUE
      CALL CONCCO(GCO,EFFCON)
C     ALTHOUGH SOME ARRAYS ARE UPDATED BY PREVIOUS CALL, WE MUST ALSO
C        UPDATE:  ESS1S,LISTL,POINTL,LISTC,POINTC
C     UPDATE LISTC,POINTC
      Y = POINTC
      DO 152 I=1,Y
      IF(LISTC(I).NE.CCO)GO TO 152
      POINTC = POINTC - 1
      IF(I.GT.POINTC) GO TO 153
      DO 154 J=I,POINTC
      LISTC(J) = LISTC(J+1)
  154 CONTINUE
  152 CONTINUE
  153 CONTINUE
C     UPDATE LISTL,POINTL
      CALL ORDERL(GCO)
      GO TO 18
C
C     SECTION 32 - UPDATE GSMALL'S FOR THOSE GATES STILL CONNECTED
C        TO GCO
C
C     FORCE NECESSARY ZEROS IN GSMALL
   32 X = LIPRED(GCO)
      DO 157 J=1,X
      Y = IPRED(J,GCO)
      DO 158 II=1,F$UB0
      I = F$0(II)
      GSMALL(Y,I) =-100
  158 CONTINUE
  157 CONTINUE
C     CHOOSE NECESSARY ONES ACCORDING TO RORDER
      POINT = 0
      DO 170 I = 1,NR
      Y = RORDER(I)
      IF(INC$MX(Y,GCO).EQ.0) GO TO 170
      POINT = POINT + 1
      ORDERP(POINT) = Y
  170 CONTINUE
C     NOTE NOW POINT = LIPRED(GCO)
      DO 159 KK=1,F$UB1
      K = F$1(KK)
      GOTO(11111,22222,33333,44444,55555),JFLAG
C     VERSION 1 :    1) NEW    2) OLD
11111 PICK1 = 0
      PICK2 = 0
      DO 173 J=1,POINT
      Y = ORDERP(J)
      IF(P$(2,(Y-1)*N2+K).NE.1) GO TO 173
      IF(GSMALL(Y,K).GE.1) GO TO 159
```

```
      IF(PICK1.NE.0) GO TO 173
      IF(PICK2.EQ.0) PICK2 = Y
      IF(ORIGIN(Y).EQ.0) PICK1 = Y
  173 CONTINUE
      IF(PICK1.EQ.0) GO TO 188
      GSMALL(PICK1,K) = 1
      GO TO 159
  188 GSMALL(PICK2,K) = 1
      GO TO 159
C     VERSION 2 :   1) NG + NEV   2) OEV   3) OG
22222 PICK1 = 0
      PICK2 = 0
      PICK3 = 0
      DO 175 J = 1,POINT
      Y = ORDERP(J)
      IF(P$(2,(Y-1)*N2+K).NE.1) GO TO 175
      IF(GSMALL(Y,K).GE.1) GO TO 159
      IF(PICK1.NE.0) GO TO 175
      IF(ORIGIN(Y).NE.0) GO TO 207
      PICK1 = Y
      GO TO 175
  207 IF(PICK2.NE.0) GO TO 175
      IF(Y.GT.N) GO TO 208
      PICK2 = Y
      GO TO 175
  208 IF(PICK3.EQ.0) PICK3 = Y
  175 CONTINUE
      IF(PICK1.EQ.0) GO TO 176
      GSMALL(PICK1,K) = 1
      GO TO 159
  176 IF(PICK2.EQ.0) GO TO 209
      GSMALL(PICK2,K) = 1
      GO TO 159
  209 GSMALL(PICK3,K) = 1
      GO TO 159
C     VERSION 3 :   1) NEV   2) OEV   3) NG   4) OG
33333 PICK1 = 0
      PICK2 = 0
      PICK3 = 0
      PICK4 = 0
      DO 177 J=1,POINT
      Y = ORDERP(POINT+1-J)
      IF(P$(2,(Y-1)*N2+K).NE.1) GO TO 177
      IF(GSMALL(Y,K).GE.1) GO TO 159
      IF(Y.GT.N) GO TO 178
      IF(ORIGIN(Y).NE.0) GO TO 179
      PICK1 = Y
      GO TO 177
  179 PICK2 = Y
      GO TO 177
  178 IF(ORIGIN(Y).NE.0) GO TO 210
      PICK3 = Y
      GO TO 177
  210 PICK4 = Y
  177 CONTINUE
      CHOICE = PICK1
      IF(CHOICE.NE.0) GO TO 211
      CHOICE = PICK2
      IF(CHOICE.NE.0) GO TO 211
      CHOICE = PICK3
      IF(CHOICE.NE.0) GO TO 211
```

```
      CHOICE = PICK4
  211 GSMALL(CHOICE,K) = 1
      GO TO 159
C     VERSION 4 :   1) NEV   2) NG   3) OEV   4) OG
44444 PICK1 = 0
      PICK2 = 0
      PICK3 = 0
      PICK4 = 0
      DO 180 J = 1,POINT
      Y = ORDERP(POINT+1-J)
      IF(P$(2,(Y-1)*N2+K).NE.1) GO TO 180
      IF(GSMALL(Y,K).GE.1) GO TO 159
      IF(Y.GT.N) GO TO 181
      IF(ORIGIN(Y).NE.0) GO TO 182
      PICK1 = Y
      GO TO 180
  182 PICK3 = Y
      GO TO 180
  181 IF(ORIGIN(Y).NE.0) GO TO 212
      PICK2 = Y
      GO TO 180
  212 PICK4 = Y
  180 CONTINUE
      CHOICE = PICK1
      IF(CHOICE.NE.0) GO TO  213
      CHOICE = PICK2
      IF(CHOICE.NE.0) GO TO 213
      CHOICE = PICK3
      IF(CHOICE.NE.0) GO TO 213
      CHOICE = PICK4
  213 GSMALL(CHOICE,K) = 1
      GO TO 159
C     VERSION 5 :   1) EV   2) G
55555 PICK1 = 0
      PICK2 = 0
      DO 183 J = 1,POINT
      Y = ORDERP(POINT+1-J)
      IF(P$(2,(Y-1)*N2+K).NE.1) GO TO 183
      IF(GSMALL(Y,K).GE.1) GO TO 159
      PICK2 = Y
      IF(Y.LE.N)PICK1 = Y
  183 CONTINUE
      CHOICE = PICK2
      IF(PICK1.NE.0) CHOICE = PICK1
      GSMALL(CHOICE,K) = 1
      GO TO 159
  159 CONTINUE
      GO TO 5
C
C     SECTION 7  -  TRY TO REMOVE CONNECTIONS PREVIOUSLY ADDED
C
    7 CALL SUBNET
      CALL PVALUE
      CALL MINI2(IMPROV)
C     CALCULATE NUMBERS OF GATES AND CONNECTIONS IN NEW NETWORK
      GAFTER = M
      CAFTER = 0
      DO 165 I=1,NR
      CAFTER = CAFTER + LISUCC(I)
      IF(I.LE.NM) GO TO 165
      IF(LISUCC(I).GT.0) GAFTER = GAFTER + 1
```

```
  165 CONTINUE
C      COST OF NEW NETWORK IS :
       NEWCST = GAFTER*A + CAFTER*B
C      COST OF ORIGINAL NETWORK WAS :
       OLDCST = GBEFOR*A + CBEFOR*B
C      SO COST SAVINGS OF TRANSFORMED NETWORK IS:
       SAVING = OLDCST - NEWCST
       PRINT 8963,SAVING
 8963  FORMAT(' SAVING = ',I5)
C
C      SECTION 8  -  IMPROVED NETWORK OBTAINED ?
C
    8  IF(SAVING.GE.0)GO TO 9
       GO TO 14
C
C      SECTION 9  -  IMPROVED NETWORK, SO UPDATE ARRAYS
C
C      UPDATE COST
C      COST = COST - SAVING
    9  GO TO 10
C
C      SECTION 14  -  RESTORE  ORIGINAL NETWORK
C
C      RESTORE INC$MX FROM QINC$M
   14  DO 171 I=1,NR
       DO 171 J=1,NR
  171  INC$MX(I,J) = QINC$M(I,J)
       CALL SUBNET
       CALL PVALUE
       GO TO 10
C
C      SECTION 10  -  END OF PROCEDURE II  (PROCII)
C
   10  RETURN
       END


       SUBROUTINE RNONES(GCO,TH,        FLAG)
C      EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C
C      THIS SUBROUTINE PERFORMS NECESSARY UPDATES WHEN AN INPUT TH IS
C         REMOVED FROM A GATE GCO.  ARRAY "ESSIS" IS ONLY PARTIALLY
C         UPDATED (SO MUST BE COMPLETED BY CALLING SECTION OF PROGRAM).
C         (RNONES = REMOVE NON-ESSENTIAL INPUT)
C
C      IF FLAG = 0, SKIP UPDATE OF LISTL,POINTL,INPTCV
C      IF FLAG=1, SKIP LISTL UPDATE, IF FLAG=2, DO NOT SKIP
C
       IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C      DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C      I,J,X ARE USED AS JUST TEMPORARY VARIABLES.
C
       COMMON NERMAX
       COMMON    N            , M            , A            , B
     1      ,    R            , N2           , N1           , NR
     2      ,    NM           , KFLAG        , JFLAG        , COST
     3      ,    LEVM         , NRN2         , NM1          , NN2
       COMMON    ISUCC(40,40) , LISUCC(40)   , IPRED(40,40) , LIPRED(40)
     1      ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)   , UNAME(40)
```

```
     2      ,   GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)  , TIME
      COMMON    T             , RTCONN(100)   , S             , RSCONN(100)
      COMMON    IFLAG         ,POINTA         ,ESS1S(40)      ,F$1(32)
     1     ,F$UB1             ,INPTCV(32)     ,LISTC(40)      ,POINTC
     2     ,LISTL(40)         ,POINTL         ,ORIGIN(40)     ,IPATH(40)
     3     ,POINTR            ,VF$1(32)       ,VF$UB1         ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2)
     1     ,RPLC(2,40)        ,IDX0(32)       ,IDX0E(32)      ,IDX1(32)
     2     ,IDX1E(32)         ,SUMP(32)       ,SETT1(32)      ,NOT1
     3     ,SETS1(40)         ,NOS1           ,SETS(40)       ,NOS
     4     ,STS               ,SUMS2(32)      ,SETS2(200)     ,NOS2
     5     ,LIP               ,NOOE           ,KEYA           ,KEYB
     6     ,NOO               ,NO1            ,NO1E           ,$GT
     7     ,$LTH              ,$PW            ,$NOE           ,GI
      COMMON                  NOT1SV         ,NOS1SV         ,LMTS2
C     UPDATE INC$MX
      INC$MX(TH,GCO)=0
C     UPDATE LISUCC AND ISUCC
      X = LISUCC(TH)
      DO 93 I=1,X
      IF(ISUCC(I,TH).EQ.GCO)GOTO94
   93 CONTINUE
   94 X = LISUCC(TH) - 1
      LISUCC(TH) = X
      IF(I.GT.X)GOTO129
      DO 95 J=I,X
      ISUCC(J,TH)=ISUCC(J+1,TH)
   95 CONTINUE
C     UPDATE LIPRED AND IPRED
  129 X = LIPRED(GCO)
      DO 98 I=1,X
      IF(IPRED(I,GCO).EQ.TH)GO TO    8
   98 CONTINUE
    8 X=X - 1
      LIPRED(GCO) = LIPRED(GCO) - 1
      IF(I.GT.X)GOTO92
      DO 99 J=I,X
      IPRED(J,GCO)=IPRED(J+1,GCO)
   99 CONTINUE
C     UPDATE SUC$MX
   92 CALL SUCCES
      IF(FLAG.EQ.0)GOTO177
C     UPDATE LISTL AND POINTL
      IF(FLAG.EQ.1)GOTO128
      DO 105 I=1,POINTL
      IF(LISTL(I).EQ.TH)GO TO 107
  105 CONTINUE
  107 POINTL= POINTL - 1
      DO 108 J=I,POINTL
      LISTL(J)=LISTL(J+1)
  108 CONTINUE
C     UPDATE INPTCV
  128 X = (TH-1)*N2
      DO 110 I=1,F$UB1
      IF(P$(2,X+F$1(I)).LE.0)GOTO110
      INPTCV(F$1(I))=INPTCV(F$1(I))-1
  110 CONTINUE
C     RECORD THE DISCONNECTION
  177 RETURN
      END
```

```
      SUBROUTINE SUBNET
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N              , M              , A              , B
     1    ,     R              , N2             , N1             , NR
     2    ,     NM             , KFLAG          , JFLAG          , COST
     3    ,     LEVM           , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1    ,     INC$MX(40,40), SUC$MX(40,40), P$(2,1280)     , UNAME(40)
     2    ,     GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
     1      ,F$UB1             ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2      ,LISTL(40)         ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3      ,POINTR            ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)        ,LPOTAB(40)      ,NRPLC(2)
     1      ,RPLC(2,40)        ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2      ,IDX1E(32)         ,SUMP(32)        ,SETT1(32)       ,NOT1
     3      ,SETS1(40)         ,NOS1            ,SETS(40)        ,NOS
     4      ,STS               ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5      ,LIP               ,NOOE            ,KEYA            ,KEYB
     6      ,NOO               ,NO1             ,NO1E            ,$GT
     7      ,$LTH              ,$PW             ,$NOE            ,G$$$$$
      COMMON                   NOT1SV          ,NOS1SV          ,LMTS2
      DIMENSION X(40),LX(40,2),OUTO(40)
C     ENTRY PRESUC
    1 CONTINUE
      DO 10 GI=1,NR
       LS=0
       LP=0
       DO 5 GJ=1,NR
        IF(INC$MX(GI,GJ).EQ.0) GO TO 3
         LS=LS+1
         ISUCC(LS,GI)=GJ
         GO TO 5
    3   IF(INC$MX(GJ,GI).EQ.0) GO TO 5
         LP=LP+1
         IPRED(LP,GI)=GJ
    5  CONTINUE
       LISUCC(GI)=LS
       LIPRED(GI)=LP
   10 CONTINUE
C
      ENTRY SUCCES
      DO 21 GI=1,NR
      DO 21 GJ=1,NR
       SUC$MX(GI,GJ)=0
   21 CONTINUE
      DO 30 GJ=N1,NR
      DO 22 GS=1,NR
        X(GS)=0
   22  CONTINUE
       X(GJ)=1
       LO=1
       LX(1,1)=GJ
       V=1
   23  CONTINUE
```

```
      V=1-V
      SWO=1+V
      SW1=2-V
      L1=0
      DO 28 LL=1,LO
       GM=LX(LL,SWO)
       LIP=LIPRED(GM)
       IF(LIP.EQ.0) GO TO 28
       DO 26 LP=1,LIP
        GP=IPRED(LP,GM)
        IF(X(GP).GT.0) GO TO 26
         SUC$MX(GP,GJ)=1
         L1=L1+1
         LX(L1,SW1)=GP
         X(GP)=1
   26   CONTINUE
   28  CONTINUE
      IF(L1.EQ.0) GO TO 30
      LO=L1
      GO TO 23
   30 CONTINUE
C
C     ENTRY LEVEL
      DO 40 GJ=1,NR
       OUTO(GJ)=LISUCC(GJ)
       GLEVEL(GJ)=-1
   40  CONTINUE
      LEV=0
   45 LEV=LEV+1
      G=0
      DO 50 GJ=1,NR
       IF(OUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50
       G=G+1
       HLIST(G,LEV)=GJ
       GLEVEL(GJ)=LEV
   50 CONTINUE
      IF(G.EQ.0) RETURN
      LGLIST(LEV)=G
      DO 60 GG=1,G
       GJ=HLIST(GG,LEV)
       LIP=LIPRED(GJ)
       IF(LIP.EQ.0) GO TO 60
       DO 55 LP=1,LIP
        GP=IPRED(LP,GJ)
        OUTO(GP)=OUTO(GP)-1
   55  CONTINUE
   60 CONTINUE
      LEVM=LEV
      GO TO 45
C
C
C
      ENTRY PVALUE
      DO 100 L=NN2,NRN2
       P$(1,L)=1
  100 CONTINUE
C
      LEV=LEVM
  110 CONTINUE
      LO=LGLIST(LEV)
      DO 130 L=1,LO
```

```
      GI=HLIST(L,LEV)
      LIS=LISUCC(GI)
      BSGI=(GI-1)*N2
      LJTH=0
      DO 115 JTH=1,N2
       IF(P$(1,BSGI+JTH).EQ.0) GO TO 115
       LJTH=LJTH+1
       X(LJTH)=JTH
  115 CONTINUE
      IF(LJTH.EQ.0) GO TO 130
      DO 125 LS=1,LIS
       GS=ISUCC(LS,GI)
       BSGS=(GS-1)*N2
       DO 120 LJ=1,LJTH
        P$(1,X(LJ)+BSGS)=0
  120  CONTINUE
  125 CONTINUE
  130 CONTINUE
      LEV=LEV-1
      IF(LEV.GE.2) GO TO 110
      RETURN
C
C
C
      ENTRY RSTRCT(KEYRST)
      KEYRST=0
      IF(LEVM.GT.LMAX)GO TO 160
      DO 150 GI=N1,NR
       IF(LIPRED(GI).GT.FANIN)GO TO 160
       IF(LISUCC(GI).GT.FANOUT)GO TO 160
  150 CONTINUE
      RETURN
  160 KEYRST=1
      RETURN
      ENTRY UNNECE
C***** THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH
C       TO OUTPUT GATES *****
      TS=T
      DO 209 GI=NM1,NR
       IF(GLEVEL(GI).EQ.1) GO TO 207
       DO 205 GJ=N1,NM
        IF(SUC$MX(GI,GJ).GT.0) GO TO 209
  205  CONTINUE
C***** GI IS REDUNDANT *****
  207 CONTINUE
       LIP=LIPRED(GI)
       IF(LIP.EQ.0) GO TO 206
       DO 203 LI=1,LIP
        GK=IPRED(LI,GI)
      IF(INC$MX(GK,GI).LE.0) GO TO 203
        T=T+1
        RTCONN(T)=100*GK+GI
        INC$MX(GK,GI)=0
  203  CONTINUE
  206  LIS=LISUCC(GI)
       IF(LIS.EQ.0) GO TO 209
       DO 204 LI=1,LIS
        GK=ISUCC(LI,GI)
      IF(INC$MX(GI,GK).LE.0) GO TO 204
        T=T+1
        RTCONN(T)=100*GI+GK
```

```
      INC$MX(GI,GK)=0                                                   G1 20340
204   CONTINUE                                                          G1 20350
209 CONTINUE                                                            G1 20360
    IF(T.GT.TS) GO TO 1                                                 G1 20370
    RETURN                                                              G1 20380
    END                                                                 G1 20390
```

```
C
C
C*****************************************************************************
C                                                                            *
C                                                                            *
C          PPPP     RRRR     OOO     GGG     RRRR      A      M     M        *
C          P   P    R   R   O   O   G   G    R   R    A A     MM   MM        *
C          P   P    R   R   O   O   G        R   R   A   A    M M M M        *
C          PPPP     RRRR    O   O   G  GG    RRRR    AAAAA    M  M  M        *
C          P        R  R    O   O   G   G    R  R    A   A    M     M        *
C          P        R   R    OOO     GGG     R   R   A   A    M     M        *
C                                                                            *
C                                                                            *
C                                                                            *
C N   N    EEEEE    TTTTT    TTTTT   RRRR      A             GGG     222     *
C NN  N    E          T        T     R   R    A A           G   G   2   2    *
C N N N    E          T        T     R   R   A   A          G          2     *
C N  NN    EEE        T        T     RRRR    AAAAA   XXXXX  G  GG      2     *
C N   N    E          T        T     R  R    A   A          G   G    2       *
C N   N    EEEEE      T        T     R   R   A   A           GGG    22222    *
C                                                                            *
C                                                                            *
C*****************************************************************************
C
C


      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C        $GT: POINTS TO A 'COLUMN' OF PCTAB. FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C       $LTH: POINTS TO A 'COLUMN' OF PCTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C       $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C             ROW IS USED.
C        $PW: POINTS TO A 'COLUMN' OF PCTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C          A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C          B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C       COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C      ESSIS: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C             (POSITIONS IN ESSIS CORRES. TO GATES NOT FEEDING GCO ARE
C             IGNORED).
C      FSUB1: POINTS TO LAST ELEMENT IN F$1.
C        F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C             COVERING) IN A CONNECTIBLE FUNCTION.
C         GI: LABEL OF A PARTICULAR GATE.
C     GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C     GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C      HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C             WORK LEVEL J.
C       IDXO: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C      IDXOE: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C       IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
```

```
C        IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C               CONSIDERATION.
C        IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.
C       INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE
C               (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.
C       INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTS
C               HAVE A '1' IN THE POSITION INDICATED BY F$1.
C        IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE
C               TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.
C        IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. IN
C               A LIST OF GATES AND EX. VAR. FEEDING GJ.
C        ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.
C        JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.
C         KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN
C               PERFORMED.
C         KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS
C               BEEN COMPENSATED.
C        KFLAG: SAME AS KEIFLG IN PROCII.
C         LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO
C               ASSIGNED LEVELS JUST LIKE GATES).
C       LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF
C               NETWORK.
C          LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.
C       LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.
C        LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY
C               DECREASING NO. OF 0'S IN GCO COVERED.
C        LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED
C               GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY
C               DECREASING NO. OF ESSENTIAL 1'S.
C       LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR
C               EX. VAR.) GI.
C        LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.
C       LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB
C               CONCERNING GI.
C            M: NUMBER OF NETWORK OUTPUT GATES.
C            N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.
C       NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX
C               ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,
C               PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S
C               REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL
C               TO ONE HALF OF N2 BY DEFAULT.
C           NM: SUM OF N PLUS M
C          NM1: SUM OF NM PLUS 1.
C          NN2: PRODUCT OF N AND N2.
C          NOS: NUMBER OF ELEMENTS IN SET S.
C         NOS1: NUMBER OF ELEMENTS IN SET S1.
C       NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE
C               RPLCF.
C         NOS2: NUMBER OF ELEMENTS IN SET S2.
C         NOT1: NUMBER OF ELEMENTS IN SET T1.
C       NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE
C               RPLCF.
C          NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.
C         NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.
C          NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.
C         NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.
C           NR: SUM OF N PLUS R.
C         NRN2: PRODUCT OF NR AND N2.
C        NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)
C                                                     FOR I=1,2.
C           N1: SUM OF N PLUS 1.
```

```
C          N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED
C              (USUALLY 2 TO THE POWER N).
C      ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.
C              ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.
C          P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND
C              EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),...,
C              P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR
C              SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR
C              FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR
C              CALCULATIONS ASSOCIATED WITH P$(1,-).
C         PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE
C              REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED
C              NETWORK.
C      POINTA: NOT USED.
C      POINTC: POINTS TO LAST ELEMENT IN LISTC.
C      POINTL: POINTS TO LAST ELEMENT IN LISTL.
C      POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).
C       POTAB: POSSIBLE OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL
C              COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY
C              USEFUL) FUNCTIONS.
C      PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF
C              ROWS OF POTAB CONCERNING GI.
C           R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO
C              NOTE SOME OF R GATES MAY BE ISOLATED).
C        RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                        ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.
C              RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                        AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.
C      RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).
C      RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).
C           S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST
C              ENTRY IN RSCONN.
C        SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATION
C              WHICH ARE TO BE REPLACED IF POSSIBLE.
C       SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE
C              REPLACED BY ELEMENTS IN SET S2.
C       SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR
C              REPLACING ELEMENTS IN SET S.
C       SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS IN
C                                                             SET S1.
C         STS: STARTING ELEMENT OF SET S.
C      SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.
C              SUC$MX(GI,GJ)=0 IF NOT.
C        SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.
C       SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.
C           T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO
C              LAST ENTRY IN RTCONN.
C        TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.
C       UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.
C     VF$UB1: POINTS TO LAST ELEMENT IN VF$1.
C        VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONS
C              (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING
C              ORIGINALLY CONNECTED INPUTS TO GCO.
C
C
C
      COMMON NEPMAX
      COMMON    N              , M              , A              , B
     1     ,    P              , N2             , N1             , NR
     2     ,    NM             , KFLAG          , JFLAG          , COST
     3     ,    LEVM           , NRN2           , NM1            , NN2
```

```
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40) , LIPRED(40)  G2 01570
     1      ,   INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)   G2 01580
     2      ,   GLEVEL(40)   , LGLIST(40)    , HLIST(40,40) , TIME        G2 01590
      COMMON    T            , RTCONN(100)   , S            , RSCCNN(100) G2 01600
      COMMON    IFLAG        ,POINTA         ,ESS1S(40)     ,F$1(32)      G2 01610
     1     ,F$UB1            ,INPTCV(32)     ,LISTC(40)     ,POINTC       G2 01620
     2     ,LISTL(40)        ,POINTL         ,ORIGIN(40)    ,IPATH(40)    G2 01630
     3     ,POINTR           ,VF$1(32)       ,VF$UB1        ,GSMALL(40,32)G2 01640
      COMMON    POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)    ,NRPLC(2)     G2 01650
     1     ,RPLC(2,40)       ,IDX0(32)       ,IDX0E(32)     ,IDX1(32)     G2 01660
     2     ,IDX1E(32)        ,SUMP(32)       ,SETT1(32)     ,NCT1         G2 01670
     3     ,SETS1(40)        ,NOS1           ,SETS(40)      ,NOS          G2 01680
     4     ,STS              ,SUMS2(32)      ,SETS2(200)    ,NOS2         G2 01690
     5     ,LIP              ,NOOE           ,KEYA          ,KEYB         G2 01700
     6     ,NOO              ,NO1            ,NO1E          ,$GT          G2 01710
     7     ,$LTH             ,$PW            ,$NOE          ,GI           G2 01720
      COMMON                  NOT1SV         ,NOS1SV        ,LMTS2        G2 01730
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)                           G2 01740
      DATA KOUNT5 /0/, UBLANK/'    '/                                     G2 01750
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX               G2 01760
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERRCR POSITICNS           G2 01770
 1000 FORMAT(20A4/5I4,A4,I4)                                              G2 01780
      KEYXC=0                                                             G2 01790
      IF(UC.NE.UBLANK) KEYXC=1                                            G2 01800
      CALL PAGE                                                           G2 01810
      CALL LINE(10)                                                       G2 01820
      KOUNT5=KOUNT5+1                                                     G2 01830
      PRINT 2, KOUNT5                                                     G2 01840
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 )   G2 01850
      CALL LINE(4)                                                        G2 01860
      PRINT 1005, UHEAD                                                   G2 01870
 1005 FORMAT(25X,20A4)                                                    G2 01880
      CALL LINE(4)                                                        G2 01890
      PRINT 10, N,M,A,B                                                   G2 01900
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //                            G2 01910
     1        30X,'NUMBER OF FUNCTIONS =',I4 //                           G2 01920
     2        30X,'COST COEFFICIENT A  =',I4//                            G2 01930
     3        47X,                'B   =',I4)                             G2 01940
      CALL LINE(1)                                                        G2 01950
      IF(KEYXC.NE.0) GO TO 25                                             G2 01960
      PRINT 21                                                            G2 01970
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')               G2 01980
      GO TO 30                                                            G2 01990
   25 CONTINUE                                                            G2 02000
      PRINT 28                                                            G2 02010
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCCMPLEMENTED VARIABLESG2 02020
     1 X, Y ---')                                                         G2 02030
   30 CONTINUE                                                            G2 02040
      CALL LINE(5)                                                        G2 02050
C***** SET UP EXTERNAL VARIABLES *****                                    G2 02060
      N2=2**N                                                             G2 02070
      IF(NEPMAX.EQ.0)NEPMAX = N2/2                                        G2 02080
      H=N*N2                                                              G2 02090
      J=N2                                                                G2 02100
      L= 1                                                                G2 02110
      I=0                                                                 G2 02120
      DO 1011 II=1,N                                                      G2 02130
       J=J/2                                                              G2 02140
       L=L*2                                                              G2 02150
       SN= 1                                                              G2 02160
       DO 1010 LL=1,L                                                     G2 02170
```

```
         SN=-SN
         V=(1+SN)/2
         DO 1009 JJ=1,J
          I=I+1
          P$(1,I)=V
       IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009    CONTINUE
 1010   CCNTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,    CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DO 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWCRK    COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      S = 0
      T = 0
      CALL UNNECE
      GATES = M
      C = 0
      DO 4 GI =  1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM)GOTO4
      IF(LISUCC(GI).GT.0)GATES=GATES+1
    4 CONTINUE
      OLDCST = A*GATES + B*(C)
      T=0
      S=0
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
```

```
C****   PROCEDURE  PROCIV
      CALL PROCIV(3)
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORK DERIVED BY PROCIV')
      PRINT 3916,TIME
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      GATES = M
      C = 0
      DO 36 GI = 1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM) GO TO 36
      IF(LISUCC(GI).GT.0) GATES = GATES + 1
   36 CONTINUE
      NEWCST = A*GATES + B*C
      IF(NEWCST.LT.OLDCST)GO TO 37
      PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
   37 CALL LINE(3)
      PRINT 320,NEWCST
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCIV'/9X,' COST=',I5,'.')
      GO TO 990
  500 STOP
      END


      SUBROUTINE PROCIV(VERSIN)
C     EDITION  CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
C     VARIABLE DEFINITIONS:
C         CA: NO. OF CONNECTIONS IN NETWORK AFTER CALLING MINI2.
C         CB: NO. OF CONNECTIONS IN NETWORK BEFORE CALLING MINI2.
C      CONNS: NO. OF CONNECTIONS REMOVED BY MINI2.
C     COUNTP: NUMBER OF CONSECUTIVE TIMES PROCII HAS BEEN CALLED WITHOUT
C             REMOVING A GATE.
C         GA: NO. OF GATES IN NETWORK AFTER CALLING A SUBROUTINE.
C       GATE: GATE OF FOCUS FOR APPLICATION OF PROCEDURE (I.E., PROGRAM
C             CONCENTRATES ON ITS REMOVAL).
C      GATES: NO. OF GATES REMOVED BY MINI2.
C         GB: NO. OF GATES IN NETWORK BEFORE CALLING A SUBROUTINE.
C     IMPROV: A PARAMETER RETURNED BY MINI2 INDICATING WHETHER IT WAS
C             SUCCESSFUL (=1) OR NOT (=0).
C     NEWCST: COST OF (POSSIBLY) NEW NETWORK DERIVED BY MINI2.
C     ONECNT: COUNTS NO. OF '1' COMPONENTS IN CSPF VECTOR OF SOME GATE.
C       ONES: ONES(GI) GIVES THE NUMBER OF '1' COMPONENTS IN GI'S CSPF
C             VECTOR
C     PORDER: AN ORDERING OF GATES (NO EX. VARS.) ACCORDING TO INCREAS-
C             ING NUMBERS OF 1'S IN THEIR CSPF VECTORS.
C     VERSIN: VERSION OF A PARTICULAR ORDERING TO BE USED BY PROCII IN
C             ASSIGNING COVERS.
C
```

```
C       I,J,MAX,MIN,PSUB ARE USED AS JUST TEMPORARY VARIABLES.
C
C       HOW TO INCREASE CAPACITY OF SUBROUTINE.
C       DIMENSION: ONES(X),PORDER(X) - X EQUAL TO MAX NUMBER OF GATES
C                                         PLUS EXTERNAL VARIABLES.
C
        COMMON NEPMAX
        COMMON    N              , M              , A              , B
      1      ,    P              , N2             , N1             , NR
      2      ,    NM             , KFLAG          , JFLAG          , COST
      3      ,    LEVM           , NRN2           , NM1            , NN2
        COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
      1      ,    INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
      2      ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
        COMMON    T              , RTCCNN(100)    , S              , RSCONN(100)
        COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
      1      ,F$UB1              ,INPTCV(32)      ,LISTC(40)       ,POINTC
      2      ,LISTL(40)          ,POINTL          ,ORIGIN(40)      ,IPATH(40)
      3      ,POINTR             ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
        COMMON    POTAB(200,42)  ,PPOTAB(40)      ,LPOTAB(40)      ,NRPLC(2)
      1      ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
      2      ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)       ,NOT1
      3      ,SETS1(40)          ,NOS1            ,SETS(40)        ,NOS
      4      ,STS                ,SUMS2(32)       ,SETS2(200)      ,NOS2
      5      ,LIP                ,NO0E            ,KEYA            ,KEYB
      6      ,NO0                ,NO1             ,NO1E            ,$GT
      7      ,$LTH               ,$PW             ,$NOE            ,GI
        COMMON                    NOT1SV         ,NOS1SV          ,LMTS2
C
C       'VERSIN'  INDICATES THE ORDERING TO BE USED IN CALL TO PROCII
C         VERSION = 3 OR 5 RECOMMENDED, BUT VERSION = 1,2, OR 4 ALSO GOOD
        DIMENSION ONES(40),PORDER(40)
        GB = M
        CB = 0
        DO 2 I = 1,NR
        CB = CB + LISUCC(I)
        IF(I.LE.NM) GO TO 2
        IF(LISUCC(I).GT.0) GB = GB + 1
    2   CONTINUE
    1   CALL MINI2(IMPROV)
        IF(IMPROV.NE.0) GO TO 1
        GA = M
        CA = 0
        DO 3 I = 1, NR
        CA = CA + LISUCC(I)
        IF(I.LE.NM) GO TO 3
        IF(LISUCC(I).GT.0) GA = GA + 1
    3   CONTINUE
        GATES = GB - GA
        CONNS = CB - CA
        PRINT 1000,GATES,CONNS
 1000   FORMAT(5X,I6,' GATES AND',I6,' CONNECTIONS HAVE BEEN REMOVED BY MI
      1NI2 DURING INITIALIZATION FOR PROCIV')
        IF(CONNS .EQ.0) GO TO 4
        CALL LINE(4)
        PRINT 1001
 1001   FORMAT(20X,'NETWORK DERIVED BY MINI2 AS PART OF PROCIV')
        CALL LINE(4)
        CALL TRUTH(P$,1)
        CALL LINE(4)
        CALL CKT(INC$MX,GLEVEL)
```

```
      NEWCST = A*GA + B*CA
      PRINT 1002,NEWCST
 1002 FORMAT(9X,'* A NETWORK DERIVED BY MINI2 STEP OF PROCIV'/9X,' COST
     1=',I6,'.')
C     COUNT THE NUMBER OF 1'S IN THE CSPF VECTOR FOR EACH GATE
    4 MAX = 0
      DO 5 I = N1,NR
      ONECNT = 0
      DO 6 J = 1,N2
      IF(GSMALL(I,J).LE.0)GO TO 6
      ONECNT = ONECNT + 1
    6 CONTINUE
      IF(ONECNT.GT.MAX)MAX=ONECNT
      ONES(I) = ONECNT
    5 CONTINUE
      MAX = MAX + 1
      MIN = -1
      PSUB = 1
    7 MIN = MIN + 1
      IF(MIN.EQ.MAX) GO TO 8
      DO 9 I = N1,NR
      IF(ONES(I).NE.MIN)GO TO 9
      PORDER(PSUB) = I
      PSUB = PSUB + 1
    9 CONTINUE
      GOTO7
    8 CONTINUE
      I = 1
      COUNTR = 0
C     CALCULATE GB, THE NUMBER OF GATES BEFORE CALLING PROCII
      GB = M
      DO 10 J = 1,NR
      IF(J.LE.NM) GO TO 10
      IF(LISUCC(J).GT.0) GB = GB + 1
   10 CONTINUE
   11 IF(I.GT.R) I = 1
      GATE = PORDER(I)
      IF(GATE.LE.NM) GO TO 12
      CALL PROCII(VERSIN,2,GATE,1)
C     CALCULATE GA, THE NUMBER OF GATES AFTER CALLING PROCII
      GA = M
      DO 13 J = 1,NR
      IF(J.LE.NM)GO TO 13
      IF(LISUCC(J).GT.0) GA = GA + 1
   13 CONTINUE
      IF(GA.EQ.GB) GO TO 12
      COUNTR = 0
      I = I + 1
      GB = GA
      GO TO 11
   12 COUNTR = COUNTR + 1
      I = I + 1
      IF(COUNTR.GE.R) RETURN
      GO TO 11
      END
```

```
************************************************************
*                                                          *
*                                                          *
*   NETTRA-G2 ALSO REQUIRES THE FOLLOWING SUBRCUTINES:     *
*                                                          *
*                                                          *
*   CONCCO,  ELANDO,  MINI2,  OUTPUT,  PROCII,  RNONES,    *
*                                                          *
*                      AND SUBNET                          *
*                                                          *
*                                                          *
*          (IDENTICAL TO SUBROUTINES OF THE SAME           *
*              NAMES LISTED FOR NETTRA-G1)                 *
*                                                          *
*                                                          *
************************************************************
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-698	2.	3. Recipient's Accession No.

4. Title and Subtitle PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BASED ON CONNECTABLE AND DISCONNECTABLE CONDITIONS (Reference Manual of NOR Network Transduction Programs NETTRA-G1 and NETTRA-G2)	5. Report Date February 1975
	6.
7. Author(s) J.N. Culliney	8. Performing Organization Rept. No.
9. Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801	10. Project/Task/Work Unit No.
	11. Contract/Grant No. NSF GJ-40221
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C. 20550	13. Type of Report & Period Covered Technical
	14.

15. Supplementary Notes

16. Abstracts

This paper explains the operation and usage of two FORTRAN computer programs, NETTRA-G1 and NETTRA-G2, developed for NOR-network transduction (transformation and reduction).

Existing (non-optimal) NOR-gate networks and their required output functions are given to the programs as input. The programs, in general, add, change, and/or delete connections in the network in an effort to reduce the cost of the network (defined in terms of numbers of gates and connections) as much as possible. Gates are examined individually; their input connections and potential input connections are evaluated under certain conditions of connectability and disconnectability in order to effect the changes in network configuration and thus reduce network cost.

These programs are only two out of a whole system of programs, designated by the name "NETTRA" (for NETwork TRAnsduction), which implement different NOR-network transduction procedures.

17. Key Words and Document Analysis. 17a. Descriptors

Logic design, logic circuits, logical elements, programs (computers).

17b. Identifiers/Open-Ended Terms

Computer-aided-design, permissible functions, network transduction, network transformation, redundant networks, near-optimal networks, NOR, NAND, CSPF, program manual, NETTRA-G1, NETTRA-G2.

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release unlimited	UNCLASSIFIED	
	20. Security Class (This Page) UNCLASSIFIED	22. Price